WebSphere. software

IBM

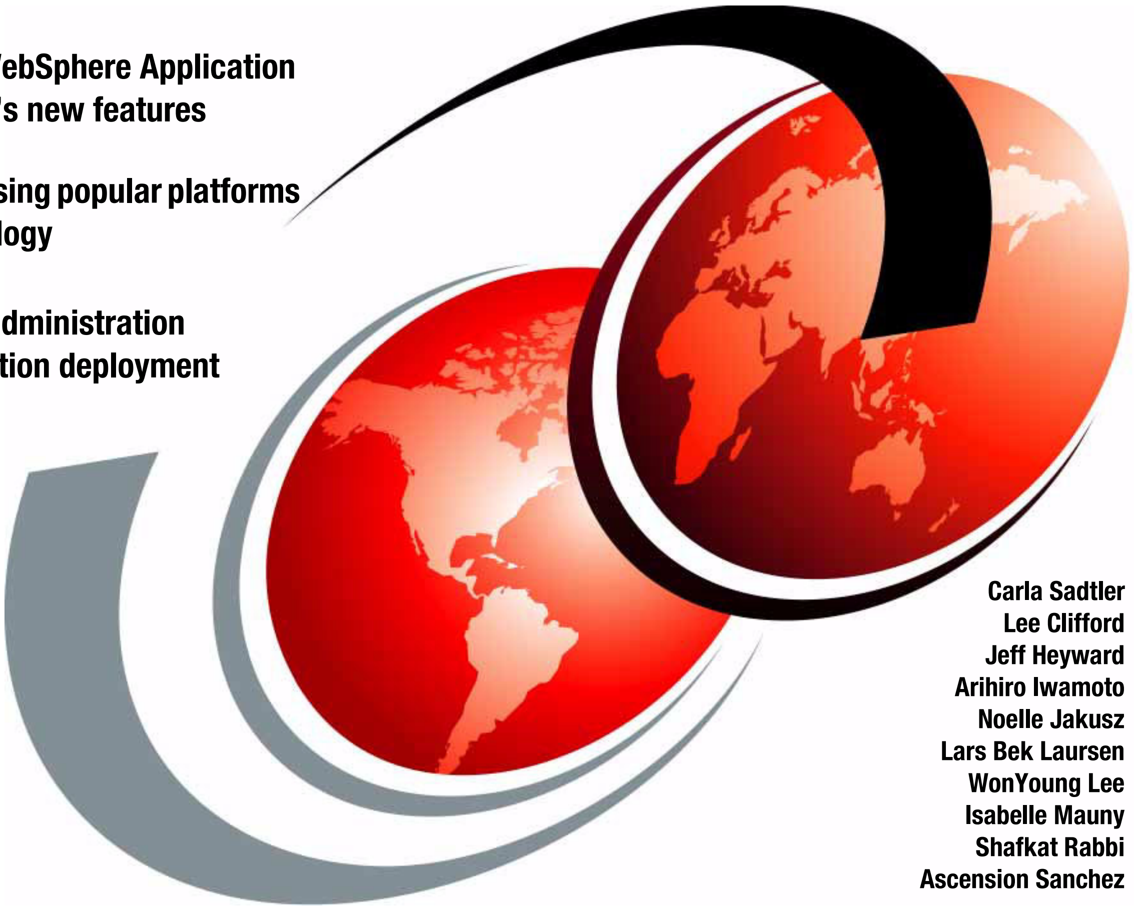
IBM WebSphere Application Server V5.1 System Management and Configuration

WebSphere Handbook Series

Exploring WebSphere Application Server V5.1's new features

Installing using popular platforms and technology

Mastering administration and application deployment



Carla Sadtler
Lee Clifford
Jeff Heyward
Arihiro Iwamoto
Noelle Jakusz
Lars Bek Laursen
WonYoung Lee
Isabelle Mauny
Shafkat Rabbi
Ascension Sanchez

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM WebSphere Application Server V5.1 System Management and Configuration WebSphere Handbook Series

April 2004

SG24-6195-01

Note: Before using this information and the product it supports, read the information in "Notices" on page xxi.

Second Edition (April 2004)

This edition applies to Version 5.1 of WebSphere Application Server and WebSphere Application Server Network Deployment.

Contents

Notices

Trademarks

Preface

This IBM® Redbook provides system administrators, developers, and architects with the knowledge needed to implement a WebSphere Application Server V5.1 Network Deployment runtime environment, to package and deploy Web applications, and to perform ongoing management of the WebSphere environment.

This is one book in a series of handbooks designed to give you in-depth information on the entire range of WebSphere Application Server products.

In this book, we provide a detailed exploration of the WebSphere Application Server V5 Base and Network Deployment runtime environments. The redbook is organized into the following parts:

- Part 1, “Getting started” on page 1. This part includes an overview of the architecture, topology options, and new features of WebSphere Application Server V5 and WebSphere Application Server Network Deployment V5.
- Part 2, “Installing WebSphere” on page 83. This part takes you through the steps needed to install each topology. Platform-specific chapters are included for installation on Windows®, AIX, and Solaris.

  For information about installing WebSphere on the Linux operating system, see *IBM WebSphere V5.0 for Linux, Implementation and Deployment Guide*, REDP3601.

  For information about installing WebSphere on the OS/400® operating system, see *WebSphere Installation, Configuration, and Administration in an iSeries Environment*, SG24-6588-00.
- Part 3, “Configuring WebSphere” on page 203. This part takes you through the process of configuring WebSphere Application Server. It is organized in the same manner as the WebSphere administrative console. It also includes information on packaging and deploying applications.
- Part 4, “Deploying applications” on page 615. This part takes you through the process of preparing and deploying an application to a WebSphere Application Server environment. It uses a sample application to illustrate how to use the Application Server Toolkit to prepare the application, then the WebSphere administrative console to deploy the application.
- Part 5, “Managing WebSphere” on page 713. This part includes information about troubleshooting runtime problems.

In addition, the handbook series consists of the following Redbooks™:

- *IBM WebSphere Application Server - Express Handbook,* SG24-6555
- *WebSphere Version 5 Web Services Handbook,* SG24-6891
- *IBM WebSphere V5.0 Security Handbook,* SG24-6573
- *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Carla Sadtler is a certified IT Specialist at the International Technical Support Organization, Raleigh Center. She writes extensively about the WebSphere and Patterns for e-business areas. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative. She holds a degree in mathematics from the University of North Carolina at Greensboro.

Lee Clifford is an IT Specialist in WebSphere Technical Sales, IBM Americas. Based in Indianapolis, IN, he works daily with customers across the central United States. He has six years of experience with IBM, having worked within both Global Services and Software. His areas of expertise include WebSphere Application Server, Portal and Host Integration. Lee holds degrees in Finance and International Studies from Indiana University.

Jeff Heyward is an IT Architect in IBM Global Services Australia. He has ten years of experience in the computing field. He holds a degree in Science (Honours) from the University of Melbourne, Australia. His areas of expertise include Java™, CORBA, object-oriented design, and WebSphere products. He also co-authored the *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176-00.

Arihiro Iwamoto is a Staff SW Engineer in WebSphere Services, Yamato Software Development Lab, Japan. He has 14 years of experience in the IT field, mainly in software development, specializing in WebSphere Application Server for the last two years. He holds a degree in Mathematics from Keio University, Japan and a Master's degree in Software Engineering from Carnegie Mellon University. His areas of expertise include Java, object-oriented design and WebSphere.

Noelle Jakusz is an IT Architect at Probitas Technologies, Inc. in Cary, NC. She currently teaches the WebSphere Application Server 4.x and 5.x Administration and Application Development curriculum through Right Source Learning

Services (an IBM Training Partner). She also teaches WebSphere Portal administration and application development. She has five years of experience in design and development of WebSphere-based applications, including WebSphere Commerce Suite and WebSphere Portal.

Lars Bek Laursen is an Advisory IT Specialist at the Integrated Technology Services division of IBM Global Services in Lyngby, Denmark. He has six years of Java experience, spanning from developing Java-based systems management solutions to designing and implementing larger application server environments. For the last three years, Lars has worked extensively as a WebSphere Application Server consultant, advising on problem solving, tuning, and implementation of fail-safe runtime environments. Lars holds a Master of Science in Engineering degree from the Technical University of Denmark.

WonYoung Lee is an Advisory IT Specialist in WebSphere Technical Sales Support, IBM Korea. He has six years of experience in Java and J2EE technologies, and extensive experience in solving critical performance problems and capacity planning issues at over 50 major customer sites. He is working on a special study of a new Performance Analysis Theory for Web-based, distributed, high-volume enterprise computing environments. He is also running a non-profit Java community, `http://www.javaservice.net`, for 20 000 enterprise-level Java developers in Korea. He holds a degree in Mathematics from Kyungpook National University, Korea.

Isabelle Mauny is a consultant for the WebSphere Software Services team for the EMEA South region, located in Madrid, Spain. She has nine years of experience in application development and object-oriented technologies. She has been working extensively with WebSphere Application Server and tools for the last three years, helping customers to design, implement, and tune J2EE applications. She also co-authored the *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176-00 and *Effective VisualAge for Java 3.5*, published by John Wiley & Sons in 2001, ISBN 0471317306.

Shafkat Rabbi is a Senior I/T Specialist working at the Professional Services in Global Services, IBM Canada. For the last few years, he has been working on customer projects involving WebSphere® technology. He has led an IBM team in assisting one of the largest Canadian banking customers to deploy WebSphere for their mission-critical Internet and intranet applications. Prior to being involved with WebSphere, his focus was providing technical services for AIX®, PSSP and HACMP™ in the e-business environment for another large Canadian bank. He has co-authored the redbook *PSSP V3 Survival Guide*, SG24-5344. He holds a Bachelor's degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology.

Ascension Sanchez is a Software Support Specialist in the e-business Support Center in Basingstoke, UK. She has five years of IT technical support experience with IBM. She holds a degree in Electronics and Electrical Engineering from the Universidad Politecnica de Madrid (Spain). Her main areas of expertise include AIX, communications software for the RS/6000®, and the WebSphere family of products for UNIX® and non-UNIX platforms. She has also co-authored the *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176-00.

Thanks to the following people for their contributions to this project:

Mark Endrei
International Technical Support Organization, Raleigh Center

Billy Newport
IBM Rochester

Logan Colby
WebSphere Development Team

Brian K Martin
WebSphere Development Team

Michael Morton
WebSphere Architect

Jason McGee
WebSphere Lead Architect

Eric Herness
WebSphere Enterprise Lead Architect

Department 7RK, WebSphere Technology and Training
IBM

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

Part 1

Getting started

This part includes an overview of the architecture, topology options, and new features of WebSphere Application Server V5 and WebSphere Application Server Network Deployment V5.

1

Introduction to IBM WebSphere Application Server

IBM WebSphere is the leading software platform for e-business on demand™. Providing comprehensive e-business leadership, WebSphere is evolving to meet the demands of companies faced with challenging business requirements, such as increasing operational efficiencies, strengthening customer loyalty, and integrating disparate systems. WebSphere provides answers in today’s challenging business environments.

IBM WebSphere is architected to enable you to build business-critical applications for the Web. WebSphere includes a wide range of products that help you develop and serve Web applications. They are designed to make it easier for customers to build, deploy, and manage dynamic Web sites more productively.

In this redbook, we take a high-level look at the following WebSphere products:

- IBM WebSphere Studio
- IBM WebSphere Application Servers
- IBM WebSphere MQ
- IBM WebSphere Portal
- IIBM WebSphere Business Integrators

1.1 WebSphere overview

WebSphere is the IBM brand of software products designed to work together to help deliver dynamic e-business quickly. WebSphere provides solutions for positively touching a customer's business. It also provides solutions for connecting people, systems, and applications with internal and external resources. WebSphere is based on infrastructure software (middleware) designed for dynamic e-business. It delivers a proven, secure, and reliable software portfolio which can provide an excellent return on investment.

The technology that powers WebSphere products is *Java*. Over the past several years, many software vendors have collaborated on a set of server-side application programming technologies that help build Web-accessible, distributed, platform neutral applications. These technologies are collectively branded as the Java 2 Platform, Enterprise Edition (J2EE) platform. This contrasts with the Java 2 Standard Edition (J2SE) platform, which most customers are familiar with, and which supports the development of client-side applications with rich graphical user interfaces (GUIs). The J2EE platform is built on top of the J2SE platform. It consists of application technologies for defining business logic and accessing enterprise resources such as databases, Enterprise Resource Planning (ERP) systems, messaging systems, e-mail servers, and so forth.

The potential value of J2EE to the customer is tremendous. Among the benefits of J2EE are the following:

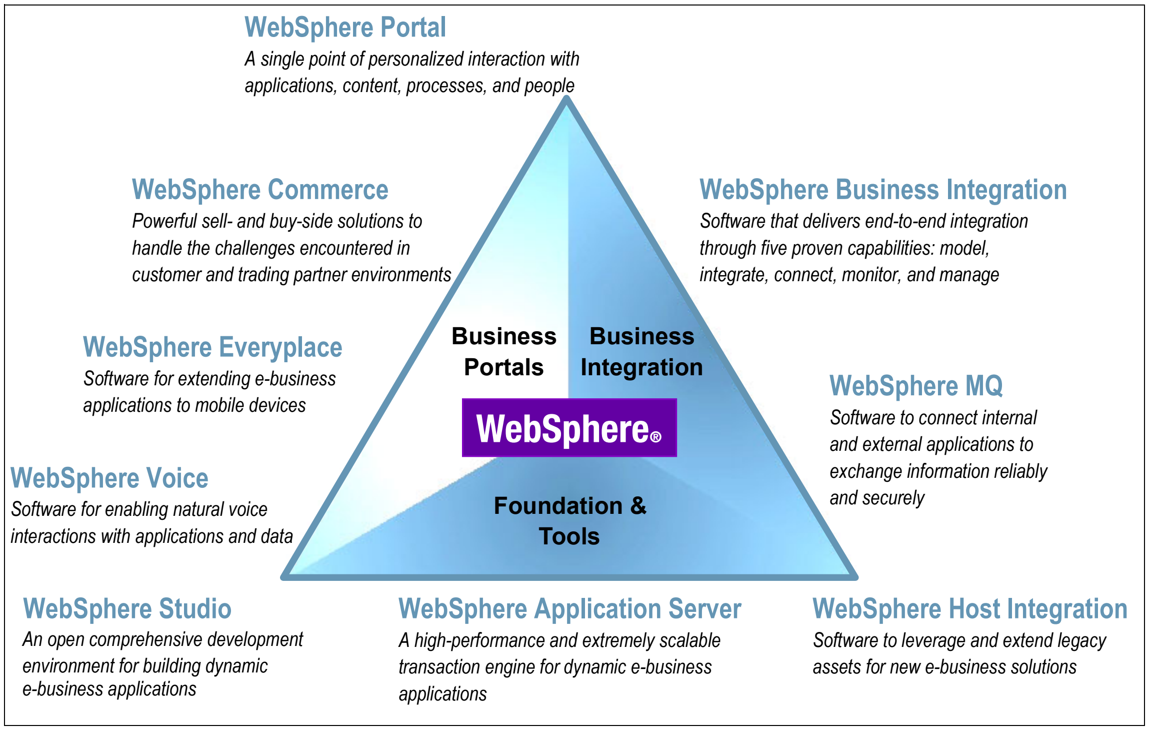
- Promotion of an architecture-driven approach to application development helps reduce maintenance costs and allows for construction of an Information Technology (I/T) infrastructure that can grow to accommodate new services.
- Application development is focused on unique business requirements and rules, rather than common application aspects, such as security and transaction support. This improves productivity and shortens development cycles.
- Industry standard technologies allow customers to choose among platforms, development tools, and middleware to power their applications.
- Embedded support for Internet and Web technologies allows for a new breed of applications that can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Another exciting opportunity for I/T is Web services. Quite simply, Web services allow for the definition of functions or services within an enterprise that can be accessed using industry standard protocols that most businesses already use today, such as HTTP and XML. This allows for easy integration of both intra- and inter-business applications, which can lead to increased productivity, expense reduction, and quicker time to market.

1.2 WebSphere family

The WebSphere platform forms the foundation of a comprehensive business solutions framework; its extensive offerings are designed to solve the problems of companies of all different sizes. For example, the technologies and tools at the heart of the WebSphere platform can be used to build and deploy the core of an international financial trading application. However, it also fits very nicely as the Web site solution for a neighborhood restaurant that simply wants to publish an online menu, hours of operation, and perhaps provide a Web-based table reservation or food delivery system. WebSphere's complete and versatile nature can sometimes be a source of confusion for people who are trying to make important decisions about platforms and developer toolkits for their business or departmental projects. Therefore, the goal of this redbook is to help you get started in understanding the technologies, tools, and offerings of the WebSphere platform.

Figure 1-1 on page 6 shows a high-level overview of the WebSphere platform.



WebSphere Portal
A single point of personalized interaction with applications, content, processes, and people
WebSphere Commerce
Powerful sell- and buy-side solutions to handle the challenges encountered in customer and trading partner environments
WebSphere Business Integration
Software that delivers end-to-end integration through five proven capabilities: model, integrate, connect, monitor, and manage
WebSphere Everyplace
Software for extending e-business applications to mobile devices
Business Portals
Business Integration
WebSphere®
WebSphere MQ
Software to connect internal and external applications to exchange information reliably and securely
WebSphere Voice
Software for enabling natural voice interactions with applications and data
Foundation & Tools
WebSphere Studio
An open comprehensive development environment for building dynamic e-business applications
WebSphere Application Server
A high-performance and extremely scalable transaction engine for dynamic e-business applications
WebSphere Host Integration
Software to leverage and extend legacy assets for new e-business solutions

Figure 1-1 WebSphere family

1.3 WebSphere foundation and tools products

Foundation and tools products reduce business risk by relying on a high quality foundation to rapidly build and deploy applications for high-performance e-business on demand. This group contains products that represent the basic functional infrastructure for other products. Some key features of these products are the following:

- **Open services infrastructure:**

  Provides a reliable foundation capable of high volume and secure transactions and is fully enabled to deliver Web services. Using an open services infrastructure allows you to provide an operating environment for disparate platforms. Creating and delivering Web services means the function only has to be developed once and can be reused anywhere it is needed. This decreases the amount of development and maintenance effort by localizing the function into one piece of code.

- **Application development:**

  Allows you to develop new applications to provide a rapid and efficient response to business needs. You can quickly build quality applications that deliver new function and integrate with existing applications and assets.

- **Enterprise modernization:**

  Allows you to leverage existing business assets and extend them for e-business. These business assets can include applications and data which provide critical business information and processes. Incorporating these existing applications into new Web-based applications can be the quickest and most cost-effective way to move to an e-business on demand environment. Leveraging your development skills is also a key part of enterprise modernization. Creating a single, consistent, interactive development environment for your entire development organization, including both Web and COBOL and PL/I developers, can provide increased skill optimization across your development organization.

Products in this category include WebSphere Application Servers, WebSphere Studio products, and WebSphere Host Integration products.

1.3.1 WebSphere Application Servers

WebSphere Application Servers are a suite of servers that implement the J2EE specification. This simply means that any Web applications that are written to the J2EE specification can be installed and deployed on any of the servers in the WebSphere Application Server family.

The foundation of the WebSphere brand is the application server. The application server provides the runtime environment and management tools for J2EE and Web services based applications.

WebSphere Application Servers are available in multiple configurations to meet specific business needs. They also serve as the base for other WebSphere products, such as WebSphere Commerce, by providing the application server required for running these specialized applications.

WebSphere Application Servers are available on a wide range of platforms, including UNIX-based platforms, Microsoft® operating systems, IBM z/OS™, and iSeries™. Although branded for iSeries, the WebSphere Application Server products for iSeries are functionally equivalent to those for the UNIX and Microsoft platforms.

Highlights and benefits

WebSphere Application Server provides the environment to run your Web-enabled e-business applications. You may think of an application server as “Web middleware,” or a middle tier in a three-tier e-business environment. The first tier is the HTTP server that handles requests from the browser client. The third tier is the business database (for example, DB2® UDB for iSeries) and the business logic (for example, traditional business applications such as order processing). The middle tier is IBM WebSphere Application Server, which provides a framework for consistent, architected linkage between the HTTP requests and the business data and logic.

IBM WebSphere Application Server is intended for organizations that want to take advantage of the productivity, performance advantages, and portability that Java provides for dynamic Web sites. It includes:

- Java runtime support for server-side Java servlets
- Support for the Enterprise JavaBeans specification
- High-performance connectors to many common back-end systems to reduce the coding effort required to link dynamic Web pages to real line-of-business data
- Application services for session and state management
- Web services that enable businesses to connect applications to other business applications, to deliver business functions to a broader set of customers and partners, to interact with marketplaces more efficiently, and to create new business models dynamically

Configurations

Because different levels of application server capabilities are required at different times as varying e-business application scenarios are pursued, WebSphere Application Server is available in five different configurations. Although they share a common foundation, each configuration provides unique benefits to meet the needs of applications and the infrastructure that supports them. So, at least one WebSphere Application Server product configuration will fulfill the requirements of any particular project and the prerequisites of the infrastructure that supports it. The following configurations are available:

- WebSphere Application Server - Express
- WebSphere Application Server
- WebSphere Application Server Network Deployment
- WebSphere Application Server Enterprise
- WebSphere Application Server for z/OS

As your business grows, the WebSphere Application Server family provides a migration path to higher configurations.

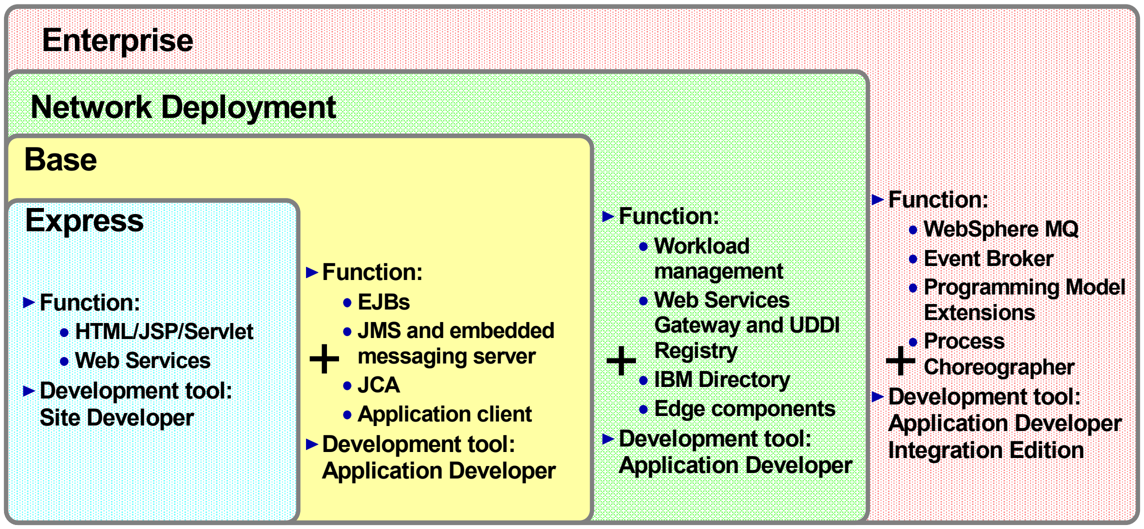

Enterprise
Network Deployment
Base
Express
Function:
HTML/JSP/Servlet
Web Services
Development tool: Site Developer
Function:
EJBs
JMS and embedded messaging server
JCA
Application client
Development tool: Application Developer
Function:
Workload management
Web Services Gateway and UDDI Registry
IBM Directory
Edge components
Development tool: Application Developer
Function:
WebSphere MQ
Event Broker
Programming Model Extensions
Process Choreographer
Development tool: Application Developer Integration Edition

Figure 1-2 WebSphere Application Server family

WebSphere Application Server - Express

The Express configuration is a low-cost, easy-to-use, out-of-the box solution that supports simple, dynamic Web sites based on the Java servlet, JavaServer Pages (JSPs), and Web services technologies. Express allows for the rapid deployment of Web applications while requiring near zero maintenance.

The Express configuration is geared toward those who need to “get started quickly” with e-business. It is specifically targeted at medium-sized businesses or departments of a large corporation, and is focused on providing ease of use and ease of application development. It provides near zero administration overhead and a one button install, and comes bundled with a very effective development tool.

Although it is easy to label WebSphere Application Server - Express as a leaner version of its more powerful siblings, it is much more than the smallest member of the family. In fact, it is an extremely capable application server that nearly implements the full J2EE specification. It only stops short of a full implementation with respect to the Enterprise JavaBean (EJB) specification. WebSphere Application Server - Express does not support EJB applications because a main design goal of the product was simplicity and ease of administration, and EJB applications tend to be more difficult to program and administer.

WebSphere Application Server - Express also contains enterprise-class connection frameworks that allow developers simple hooks into relational database systems; these connection frameworks are the same modules that ship with more advanced versions of the application server.

The WebSphere Application Server - Express offering is unique with respect to the other configurations in that it is bundled with an application development tool. Although there are WebSphere Studio configurations designed to support each WebSphere Application Server configuration, they are normally ordered and installed as independent of the server. With WebSphere Application Server - Express, you get both the server and the application development tool in a single install.

WebSphere Application Server

The WebSphere Application Server (or “base”) configuration is the next level of server infrastructure in the WebSphere Application Server family.

This configuration is for customers who want to use the full range of J2EE V1.3 technologies, including EJBs and JMS, but do not need workload management or central administration capabilities.

WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment is an even higher level of server infrastructure in the WebSphere Application Server family. It extends the base configuration to include clustering capabilities, edge services, and high availability for distributed configurations. These features become more important for larger enterprises, where applications tend to service a larger customer base, and more elaborate performance and availability requirements are in place.

Application servers in a cluster can reside on the same or multiple machines. A Web server plug-in installed in the Web server can distribute work among clustered application servers. In turn, Web containers running servlets and JSPs can distribute requests for EJBs among EJB containers in a cluster.

The addition of Edge Components provides high performance and high availability features. For example:

- The Caching Proxy intercepts data requests from a client, retrieves the requested information from the application servers, and delivers that content back to the client. It stores cachable content in a local cache before delivering it to the client. Subsequent requests for the same content are served from the local cache, which is much faster and reduces the network and application server load.
- The Load Balancer provides horizontal scalability by dispatching HTTP requests among several, identically configured Web server or application server nodes.

WebSphere Application Server Enterprise

WebSphere Application Server Enterprise provides all the features of the Network Deployment, plus programming model extensions for complex application designs. This level of application server adds sophisticated connectors for integrating disparate and legacy data sources.

An important point to make here is that, from a technology perspective, WebSphere Application Server Enterprise V5 is very far removed from the technology base we had in V3.5. This product is no longer based on Component Broker and TXSeries® technologies. It is, in fact, a set of highly valuable extensions that expand the capabilities of the core J2EE WebSphere Application Server. These extensions focus specifically on the advanced build-to-integrate capabilities and an additional set of extensions that improve the overall effectiveness of developing sophisticated applications to be run on the WebSphere Application Server platform.

It offers such capabilities as advanced application adapters, application workflow composition and choreography, extended messaging, dynamic rules-based application adaptability, internationalization, and asynchronous processing.

Features of WebSphere Application Server Enterprise include:

- Service-Oriented Architecture (SOA):

  Applications need to integrate business logic and application data within the organization and outside. SOA represents all software assets as services, including legacy applications, packaged applications, J2EE components, and Web services. It also provides standards to interact between the assets. The individual assets can be reused as building blocks in other applications.

- Integrated J2EE-based workflow:

  Using services in a business process is the next step in application development. The integrated J2EE-based workflow offers flow-based application development. In these flows, different software assets and services can be reused and composed into a business process.

  The built-in engine supports human interaction and compensation for “rollback-like” transaction support.

- Advanced transactional connectivity:

  WebSphere Application Server Enterprise provides numerous transaction support features essential for large enterprise applications, including dynamic application adapter support, last participant and ActivitySession service support, and CORBA C++ support.

- Optimized application performance:

  With application profiling techniques, applications can be fine-tuned for better performance without changing a single line of code. Network Deployment provides scalability, performance, availability, and centralized management for WebSphere Application Server Enterprise.

- Next generation development:

  WebSphere Application Server Enterprise and WebSphere Studio Application Developer Integration Edition enable “next generation” development by leveraging the latest innovations that build on today's J2EE standards, providing greater control over application development, execution, and performance than was ever possible before. These include asynchronous beans, startup beans, scheduler service, and object pools.

- Increased development productivity:

  One way to vastly improve developer productivity is to reduce the need for handcrafted solutions that can be time-consuming, costly, and difficult to maintain. WebSphere Application Server Enterprise and WebSphere Studio Application Developer Integration Edition were designed to improve developer productivity through the use of extended messaging capabilities, internationalization service, and work areas.

- Real-time application flexibility:

  In order for businesses to respond quickly to changes, applications must be able to make changes and adopt to current conditions without changing the application code. Business rule beans provide a real-time framework for defining, executing, and managing business rules that encapsulate business policies. Dynamic query allows the EJB container, using the EJB Query Language, to submit queries that select, sort, join, and perform calculations on application data at runtime.

WebSphere Application Server for z/OS

WebSphere Application Server for z/OS is specifically tailored for the z/OS platform and designed to take advantage of the z/OS and zSeries® qualities of service. WebSphere Application Server for z/OS continues to exploit the advanced quality of service capabilities on the z/OS platform, which is fully J2EE 1.3 compatible and supports the latest Web services standards.

Numerous runtime topologies are possible, ranging from a simple base configuration with one or several independent application servers up to a complex network configuration running on multiple cells defined across multiple systems in a Sysplex.

Packaging

Table 1-1 shows the features included with each WebSphere Application Server configuration.

Table 1-1 WebSphere Application Server features

	Express V5.1	**Base V5.1**	**ND V5.1**	**Enterprise V5.0.2**
Application Server	Lightweight version	X	X	X
IBM HTTP Server	embedded only	X	X	X
Web server plug-ins	X	X	X	X
Application Client		X	X	X
Application Server Toolkit		X	X	X
DataDirect Technologies JDBC Drivers for WebSphere Application Server		X	X	X
DataDirect Technologies JDBC Drivers for WebSphere Application Server - Express	X			
Deployment Manager			X	X
IBM Directory V5.1				X
IBM Directory V5.2			X	
Edge Components			X	X
Web Services Gateway			X	X
UDDI Registry			X	X
WebSphere MQ V5.3				X
WebSphere MQ Event Broker 2.1				X
WebSphere MQ for z/OS				
IBM DB2 Universal Database™ Express Edition 8.1	X			
IBM DB2 UDB Enterprise Edition 8.1		X	X	X
Programming Model Extensions				X
WebSphere Studio Site Developer	X			

	Express V5.1	Base V5.1	ND V5.1	Enterprise V5.0.2
WebSphere Development Studio Client for iSeries Version 4	iSeries only			

Application Server

The application server provides the runtime environment and management tools for J2EE and Web-services based applications.

Deployment Manager

The Deployment Manager allows you to administer multiple application servers from one centralized manager. The Deployment Manager manages all the nodes in a distributed topology and introduces the concept of application server clustering for scalability.

IBM HTTP Server

A full-function Web server that can be used as a front end to WebSphere Application Servers (via the Web server plug-in).

Web server plug-in

This provides the front-end to WebSphere Application Servers. It allows you to separate the portions of the application providing static content (HTML pages, for example) from the dynamic content (JSPs, servlets, EJBs). This allows you to place the static content in a DMZ while placing the dynamic content in an internal network. With the Network Deployment and Enterprise configurations, the plug-in provides load balancing among WebSphere Application Server clusters.

Plug-ins are provided to support the following Web servers:

- IBM HTTP Server and IBM HTTP Server for iSeries
- Apache Web Server
- Microsoft Internet Information Server (IIS)
- Lotus® Domino™ Enterprise Server
- Sun ONE Web Server, Enterprise Edition (formerly iPlanet)

IBM DB2

DB2 is a relational database product. It is included for development purposes and to provide a persistence option out of the box.

Application clients

WebSphere Application Server ships with several different types of application clients. An application client is the interface an application will use to communicate with services provided by applications running on WebSphere.

- The ActiveX application client model uses the Java Native Interface (JNI) architecture to programmatically access the Java virtual machine (JVM). Therefore, the JVM exists in the same process space as the ActiveX application (Visual Basic, VBScript, or ASP) and remains attached to the process until that process terminates.
- In the Applet application client model, a Java applet embeds in an HTML document residing on a client machine remote from the application server. With this type of client, the user accesses an enterprise bean in the application server through the Java applet in the HTML document.
- The J2EE application client is a Java application program that accesses enterprise beans, JDBC databases, and Java Message Service message queues. The J2EE application client program runs on client machines. This program follows the same Java programming model as other Java programs. However, the J2EE application client depends on the application client runtime to configure its execution environment, and uses the Java Naming and Directory Interface (JNDI) name space to access resources.
- The pluggable and thin application clients provide a lightweight Java client programming model. These clients are best suited to situations where a Java client application exists but the application needs enhancements to use enterprise beans, or where the client application requires a thinner, more lightweight environment than the one offered by the J2EE application client. The difference between the thin application client and the pluggable application client is that the thin application client includes a Java virtual machine (JVM), while the pluggable application client requires that a JVM be provided by the user.

Application Server Toolkit

This toolkit provides a number of Eclipse-based tooling products. It is a minimal footprint version of development tooling that corresponds to the base tooling provided with WebSphere Studio. The toolkit includes a debugger, a tool for application profiling, a log analyzer, and a workbench. It is installed separately from the base.

DataDirect Technologies JDBC drivers

Because J2EE-certified JDBC drivers are essential in any successful J2EE application model, DataDirect Technologies (formerly the DataDirect business unit of MERANT) creates JDBC drivers based only on the industry’s highest quality and reliability standards.

The DataDirect Technologies JDBC Drivers for WebSphere Application Server include the following drivers:

- JDBC drivers provided by a supported database
- DataDirect type 3 (SequeLink)

- DataDirect type 4 (ConnectJDBC)

IBM Directory Server V5.2

IBM Directory Server V5.2 provides a powerful Lightweight Directory Access Protocol (LDAP) identity infrastructure that is the foundation for deploying comprehensive identity management applications and advanced software architectures like Web services. LDAP V3 support ensures compatibility with industry standard LDAP-based applications.

LDAP directories are a core component of e-business infrastructures and are being integrated with many operating systems. An LDAP directory is a repository of data. The data held in an LDAP directory includes information about identity, security, applications, systems, and network management. LDAP is a standard directory schema that provides a key integration point for all network data.

WebSphere MQ V5.3 and WebSphere MQ Event Broker

The WebSphere MQ and WebSphere MQ Event Broker installation images provide the non-embedded full-function WebSphere MQ Queue Manager for reliable, dynamically load balanced asynchronous messaging for more than 35 platforms.

The included version of WebSphere MQ for z/OS cannot be used independently of WebSphere Application Server for z/OS and has significantly reduced function. A full-function WebSphere MQ for z/OS is available under a separate license.

Edge Components

The Edge Components extend WebSphere's J2EE capabilities out into the network, placing content closer to the end user. In addition to the IBM HTTP Server and Web server plug-in, the Edge Components also include:

- Load Balancer: a general purpose load balancer for dynamic routing and balancing of HTTP(s) requests among Web servers.
- Site Selector: used with the Load Balancer to act as a TCP/IP domain name server (DNS), providing increased scalability when the site load requirements are beyond the capacity of a single Load Balancer instance.
- Caching Proxy: provides SSL termination, authentication, high-speed persistent caching, and intelligent routing functions. It is used to front-end a cluster of WebSphere Application Servers. Integration with WebSphere dynamic caching provides caching of servlets and JSP pages, while integration with Load Balancer technology and WebSphere's HTTP session capabilities provides optimized workload management and routing of stateful WebSphere sessions.

Programming model extensions (PME):

The programming model extensions can be classified into the following three major groups:

- Business object model extensions operate with business objects, for example, EJBs.
- Business process model extensions provide process, workflow functionality, and services for the application server.
- Next generation applications include the rest of the extensions. They can be used in enterprise applications, where specific needs require these extensions.

Figure 1-3 shows these three major groups and the extensions that fit in them.

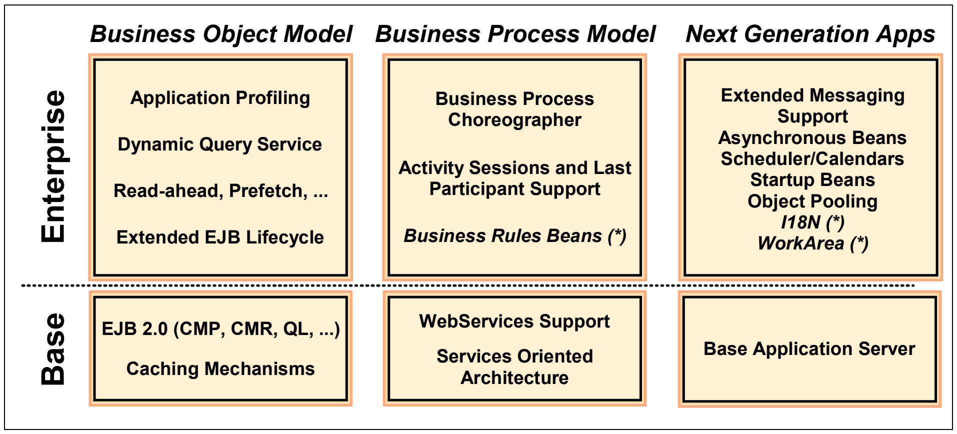


Figure 1-3 Application server functions and features in Base versus Enterprise

- Process choreographer, including staff services:

  The process choreographer is probably the most appealing extension in WebSphere Enterprise. It provides workflow and process functionality for the application server. It supports long-running and short-running, interruptible and non-interruptible processes.

  The processes are developed in WebSphere Studio Application Developer Integration Edition using a visual editor to assemble the flow. The process application is then deployed in the application server's process container.

  The runtime environment also provides a browser-based client application for administrative purposes. The default client can be freely customized and extended using the client API.

The process choreographer has an integral service called staff services, which provides advanced staff and user handling functions beyond the base user registry function.

- Extended messaging:

  Extended messaging provides significant function for building messaging-based applications. It enables you to quickly create applications that integrate with other systems through a messaging infrastructure. It offers automated support for inbound and outbound messaging by hiding the complexities of the messaging API.

  This extension provides an API built on top of JMS. In addition, complementary application development aids are provided in WebSphere Studio Application Developer Integration Edition, allowing you to build messaging components for any messaging patterns with minimum effort.

- Asynchronous beans:

  Asynchronous beans offer performance enhancements for resource-intensive tasks by enabling single tasks to run as multiple tasks. Asynchronous scheduling facilities can also be used to process parallel processing requests in “batch mode” at a designated time. WebSphere Enterprise provides full support for asynchronous execution and invocation of threads and components within the application server. The application server provides execution and security context for the components, making them an integral part of the application.

- Application profiling and access intent:

  The application profiling and access intent features provide a flexible method to fine-tune application performance for EJBs. Different EJBs or even different methods within an EJB can have their own intent to access resources. Optimizing and profiling components based on their access intent increases the performance at runtime.

- Activity sessions and last participant support:

  Last participant support is an extension to the original J2EE transaction support for applications. It allows developers to include several two-phase commit resources and one, and only one, one-phase commit resource in one unit of work.

  The activity sessions extension provides further enhancements to the J2EE session and transaction services by extending session and transaction boundaries.

- Business rule beans:

  Business rule beans externalize the business rules in an application. Rules, conditions, and decisions do not have to be hard-coded or custom tailored into applications. Business rule beans provide an API to look up and use the rules and provide an interface within the administration console to manage application rules.

- Dynamic query:

  Dynamic query fixes a weakness of Enterprise JavaBeans by enabling the client to run custom queries on EJBs during runtime. Until now, EJB lookups and field mappings were implemented at development time. Modifying the query attributes required further development. With dynamic query, the client can assemble SQL statements at runtime and execute them on any EJB. The queries can result in objects, lists of objects, or the results from aggregate functions (for example, SUM, COUNT).

- Startup beans:

  Startup beans allow the automatic execution of business logic when the application server starts or stops. For example, they might be used to pre-fill application-specific caches, initialize application-level connection pools, or perform other application-specific initialization and termination procedures.

- Scheduler service:

  The scheduler service allows you to schedule processes in a timely manner for batch processing or to maximize the utilization of existing computing resources. The scheduler service provides the ability to process workloads using parallel processing, set specific transactions as high priority, and schedule less time-sensitive tasks to process during low traffic off hours.

- Object pools:

  Application performance is essential in runtime. Object pools provide an effective means of improving performance by allowing multiple instances of objects to be reused, reducing the overhead associated with instantiating, initializing, and garbage-collecting the objects. Creating an object pool allows an application to obtain an instance of a Java object and return the instance to the pool when it has finished using it.

- Shared Work Area:

  In the process of developing software applications, passing data between application components is often a fundamental requirement. Shared Work Areas provide a solution to pass and propagate contextual information between application components.

- Internationalization (I18N) service:

  An application that can present information to users according to regional cultural conventions is said to be *internationalized*. The application can be configured to interact with users from different localities in culturally appropriate ways. The Internationalization (I18N) service provides the ability to develop internationalized applications in WebSphere.

- CORBA C++ SDK:

  The CORBA C++ SDK is used for integrating various C++ assets. This lets C++ clients invoke J2EE components using CORBA technology and also lets WebSphere applications incorporate C++ assets behind CORBA wrappers.

Web Services Gateway

The Web Services Gateway provides a framework for invoking Web services between Internet and intranet environments. Using a Web Services Gateway allows you to leverage your existing infrastructure by creating Web services as a means of communicating between systems in your enterprise.

The Web Services Gateway enables interoperability between Web services deployed on different vendor platforms. The Gateway will abstract the vendor details and publish and serve Web service requests based on the standard protocols and transports.

The Gateway is deployed as a J2EE application and can be managed by the WebSphere administrator.

IBM WebSphere UDDI Registry

Public UDDI registries allow you to publish your services for use by anyone with access to that registry. There is no validity checking done on the Web service. This kind of registry is provided to publicly distribute and utilize existing Web services.

A private registry is new to WebSphere Application Server V5. The private UDDI allows customers to become the service provider without having to comply with any additional requirements by the platform. They can use the private UDDI to publish their services directly, rather than have a third-party broker provide that service.

WebSphere Studio Site Developer

WebSphere Studio provides the development environment for applications that run on WebSphere Application Servers. As with WebSphere Application Server, it comes in multiple configurations, allowing you to select the appropriate one for your environment. In the case of WebSphere Application Server - Express, the appropriate configuration (Site Developer) is included with the package, giving you one install process.

WebSphere Application Server features

The following tables break down the highlights of support provided by each WebSphere Application Server configuration.

Table 1-2 WebSphere Application Server features

	Express V5.1	**Base V5.1**	**ND V5.1**	**Enterprise V5.0.2**
Client and server support for the Software Development Kit for Java Technology Edition 1.4 (SDK 1.4)	X	X	X	
J2EE 1.3 Programming support				
JDBC 2.0	X	X	X	X
Enterprise JavaBeans (EJB) 2.0		X	X	X
Java Servlet 2.3	X	X	X	X
JavaServer Pages (JSP) 1.2	X	X	X	X
Java Message Service (JMS) 1.0		X	X	X
Java Transaction API (JTA) 1.0		X	X	X
JavaMail 1.2	X	X	X	X
JavaBeans Activation Framework (JAF) 1.0	X	X	X	X
Java API for XML Parsing (JAXP) 1.1	X	X	X	X
J2EE Connector Architecture (JCA) 1.0	X[1]	X	X	X
RMI-IIOP	X	X	X	X
Application clients		X	X	X
J2EE 1.2 Support (EJB 1.1, Servlet 2.2, JSP 1.1)	X (no EJB support)	X	X	X
Static Web Content (HTML, GIFs, etc.)	X	X	X	X
Programming Model Extensions (Workflow, container managed messaging, and so forth)				X
Messaging support				
Embedded messaging support		X	X	X
WebSphere MQ provider		X	X	X
Generic JMS provider		X	X	X
Message-driven beans		X	X	X
Web services runtime support	X [1]	X	X	X
Security support				
Java 2		X	X	X

	Express V5.1	Base V5.1	ND V5.1	Enterprise V5.0.2
J2EE	X	X	X	X
JAAS 1.0	X	X	X	X
CSIv2		X	X	X
LDAP or local operating system user registry[3]	X	X	X	X
LTPA authentication	X	X	X	X
Simple WebSphere Authentication Mechanism (SWAM)	X	X		
Multi-node management and Edge components				
Workload management			X	X
Deployment Manager			X	X
Central administration of multiple nodes			X	X
Deployment Manager			X	X
Load Balancer			X	X
Caching Proxy			X	X
Dynamic caching		X	X	X
Performance and analysis tools				
Performance Monitoring Instrumentation (PMI)		X	X	X
Log Analyzer	Included in Studio	X	X	X
Thread analyzer		X	X	X
Tivoli® Performance Viewer		X	X	X
Administration and tools				
Web-based administration console	X	X	X	X
Assembly Toolkit		X	X	
Application Assembly Tool (AAT)				X
Integrated HTTP and Application Server Administration Console	iSeries platform			
Administrative scripting	No[2]	X	X	X
Java Management Extension (JMX) interface	X	X	X	X

	Express V5.1	Base V5.1	ND V5.1	Enterprise V5.0.2
Application Server Toolkit		X	X	X
Web services				
WS-I Basic Profile 1.0 support	X	X	X	
WS-Security	X	X	X	
SOAP 1.1	X	X	X	X
WSDL 1.1 for Web services	X	X	X	X
UDDI 2.0 for Web services	X	X	X	X
WSIL 1.0 for Web services	X	X	X	X
Web Services Gateway			X	X
Private UDDI Registry			X	X
JSR 101 (JAX-RPC)	X	X	X	X
JSR 109 (Web services for J2EE)	X	X	X	X

1. JCA support is included with Express to enable support for V5 data sources, but is not exposed.
2. The wsadmin interface for administrative scripting is included with all configurations but is for advanced users and not supported in Express.
3. LDAP support for the following servers: IBM Directory Server, Lotus Domino Enterprise Server, SecureWay® Server, Sun ONE Directory Server (formerly iPlanet), Windows 2000 Active Directory, z/OS.

Platform support

Table 1-3 shows the platform support for WebSphere Application Server by configuration.

Table 1-3 WebSphere Application Server platform support

Platform	Operating system	Express V5.1	Base V5.1	ND V5.1	Ent V5.0.2
AIX	4.3.3				X[1]
	5.1		X	X	X[1]
	5.2	Server	X	X	
HP-UX	11i				X
	11iv1	Server	X	X	

1. No Event Broker (optional) support for AIX 5.2.
2. Edge components do not run on OS/400 or z/OS.

Platform	Operating system	Express V5.1	Base V5.1	ND V5.1	Ent V5.0.2
Linux/390	Red Hat Linux for s/390 7.2				X
	SuSE SLES for s/390 7 and 8				X
	UnitedLinux V1.0		X	X	X
Linux/Intel®	Red Hat Linux Advanced Server for Intel 2.1				X
	Red Hat Linux Professional 7.2 or 8	Dev Tool			
	Red Hat Linux for Intel (x86) 8.0				X
	Red Hat Enterprise Linux WS/ES/AS for Intel (x86) 2.1	Server	X	X	X
	SuSE Linux for Intel 7.2 or 8.1	Dev Tool			
	SuSE Linux for Intel (x86) 7.3				X
	SuSE SLES for Intel 7, 8				X
	United Linux 1.0	Server	X	X	X
Linux/PPC	SuSE SLES for i/pSeries®				X
	UnitedLinux 1.0 for i/pSeries		X	X	
Sun	Solaris 8		X	X	X
	Solaris 9	Server	X	X	
OS/400	V5R1 V5R2	Server	X	X[2]	

1. No Event Broker (optional) support for AIX 5.2.
2. Edge components do not run on OS/400 or z/OS.

Platform	Operating system	Express V5.1	Base V5.1	ND V5.1	Ent V5.0.2
Windows	Windows NT® Server 4.0 SP 6a				X
	Windows 2000 Advanced Server	Server	X	X	X
	Windows 2000 Server	Server	X	X	X
	Windows Server 2003, Standard	Server	X	X	X
	Windows Server 2003 Enterprise		X	X	
	Windows Server 2003, Datacenter		X	X	
	Windows 2000 Professional	Server			X
	Windows XP Professional	Server			X
z/OS	OS/390® V2 R10 or z/OS		5.0	5.0[2]	
1. No Event Broker (optional) support for AIX 5.2. 2. Edge components do not run on OS/400 or z/OS.					

This table does not show maintenance requirements. For the exact operating system levels and requirements, see the following Web sites:

- WebSphere Application Server - Express:

  `http://www.ibm.com/software/webservers/appserv/express/requirements/`

- WebSphere Application Server:

  `http://www.ibm.com/software/webservers/appserv/was/requirements/`

- WebSphere Application Server Network Deployment:

  `http://www.ibm.com/software/webservers/appserv/was/network/requirements/`

- WebSphere Application Server Enterprise:

  `http://www.ibm.com/software/webservers/appserv/enterprise/requirements/`

- WebSphere Application Server for z/OS:

  `http://www.ibm.com/software/webservers/appserv/zos_os390/requirements/`

- WebSphere Application Server for iSeries:

  `http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/indexb50.html`

- WebSphere Application Server Network Deployment for iSeries:

  `http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/indexnd50.html`

- WebSphere Application Server - Express for iSeries:

  http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/express/indexexp50.html

1.4 What's new?

The base WebSphere Application Server and WebSphere Application Server Network Deployment have been updated since V5 was released. Two fix packs, 5.0.1 and 5.0.2, provide limited new function and minor enhancements. V5.1 provides more significant enhancements.

The highlights of WebSphere Application Server V5 are as follows:

- Full J2EE 1.3 support and support for the Java SDK 1.3.1.
- A new administrative model based on the Java Management Extensions (JMX) framework and an XML-based configuration repository. A relational database is no longer required for the configuration repository.

  A Web-based administrative console provides a GUI interface for administration.

  An interface based on the Bean Scripting Framework, *wsadmin*, has been provided for administration through scripts. In V5, the only supported scripting language is JACL.
- Security enhancements including support for Java 2, JAAS, and CSIv2.
- Common code base between z/OS and distributed platforms.

WebSphere Application Server Network Deployment V5 provides:

- Clustering, workload management, and single point of administration in a multi-node single cell topology
- Web Services Gateway
- Private UDDI Registry

WebSphere Application Server V5.0.1 has added HP-UX platform support and includes minor enhancements and fixes.

WebSphere Application Server V5.0.2 provides:

- Web services support based on JSR 101 and JSR 109. WebSphere Studio Application Developer 5.1 includes the function required to create JSR 101 and JSR 109 Web services applications.

- WS-Security integration into the WebSphere Application Server security mechanism. Web Services Gateway has also been enhanced to use WS-Security.
- Limited Tivoli Access Management support. Credential mapping using TAM V4.1 is achieved by calling the APIs directly.
- The Application Server Toolkit has been improved and now supports J2EE application assembly, logically replacing the Application Assembly Tool (AAT). The Application Server Toolkit is based on the Eclipse workbench and has the same look and feel as WebSphere Studio Application Developer.

WebSphere Application Server V5.1 provides the following enhancements:

- Java SDK 1.4.1 support. Although the new features are not yet being exploited by WebSphere, user applications have access to them.
- UDDI Registry tools that allow you to back up, restore, and modify UDDI registries.
- Web Services Gateway enhancements including JAX-RPC handler support, performance improvements, and SOAP/JMS support.
- Tivoli Access Manager V5.1 integration. Credential mapping is now done using a login module provided by Tivoli.
- Performance enhancements through optimized transaction handling.
- Support for Jython scripting language support in wsadmin.
- In a Network Deployment environment, the application server can now start without the Node Agent running.

Note: WebSphere Studio Application Developer V5.1.1 provides the support for SDK 1.4.1 and WebSphere Application Server V5.1 as a test server.

2

IBM WebSphere Application Server architecture

WebSphere Application Server is IBM's implementation of the J2EE (Java 2 Enterprise Edition) platform, conforming to V1.3 of the specification. WebSphere Application Server is available in several different configurations designed to meet a wide range of customer requirements. Each configuration is packaged with other components to provide a unique environment. At the heart of each configuration is a WebSphere Application Server which provides the runtime environment for enterprise applications. This discussion will center around the runtime server component of the five flavors of WebSphere Application Server:

- IBM WebSphere Application Server - Express V5.1. We will refer to this as the Express configuration.
- IBM WebSphere Application Server V5.1. We will refer to this as the Base configuration.
- IBM WebSphere Application Server Network Deployment V5.1. We will refer to this as the Network Deployment configuration.
- IBM WebSphere Application Server Enterprise V5.0.2. We will refer to this as the Enterprise configuration.

2.1 Server architecture

Each member of the WebSphere Application Server family uses essentially the same architectural structure. The simplest member, Express, contains a subset of the full architecture. The Base configuration is slightly more advanced; Network Deployment is even more advanced, and Enterprise is at the top.

2.1.1 Express configuration

The Express and Base configurations are similar in that they support a single-server environment. Although you can configure multiple application servers, there is no central administration or workload management. Each application server acts as an unique entity. The Express configuration is a slimmed-down version of Base. As you will see, there are several differences between the two, but perhaps the most noticeable are the absence of the EJB container and embedded messaging support in Express.

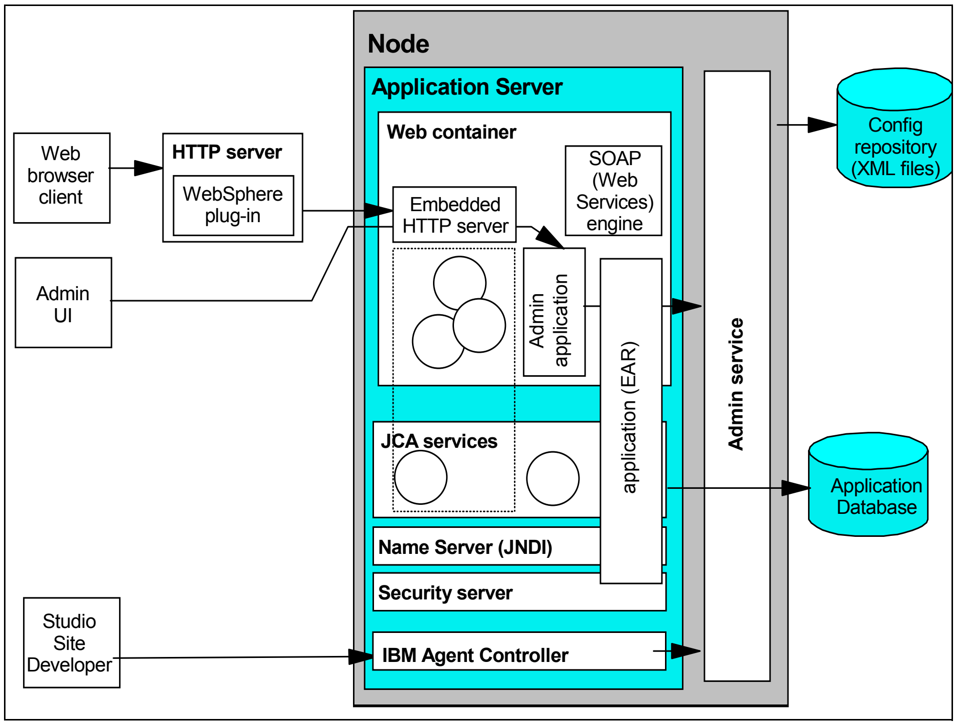


Figure 2-1 WebSphere Application Server - Express overview

2.1.2 Base configuration

The next step up in configurations is the Base configuration. Unlike the Express configuration, it features an EJB container, embedded messaging support, JCA resource adapter support and several other differences. The Base configuration has no central administration support and no workload management features.

Figure 2-2 shows an overview of the runtime architecture in a Base installation.

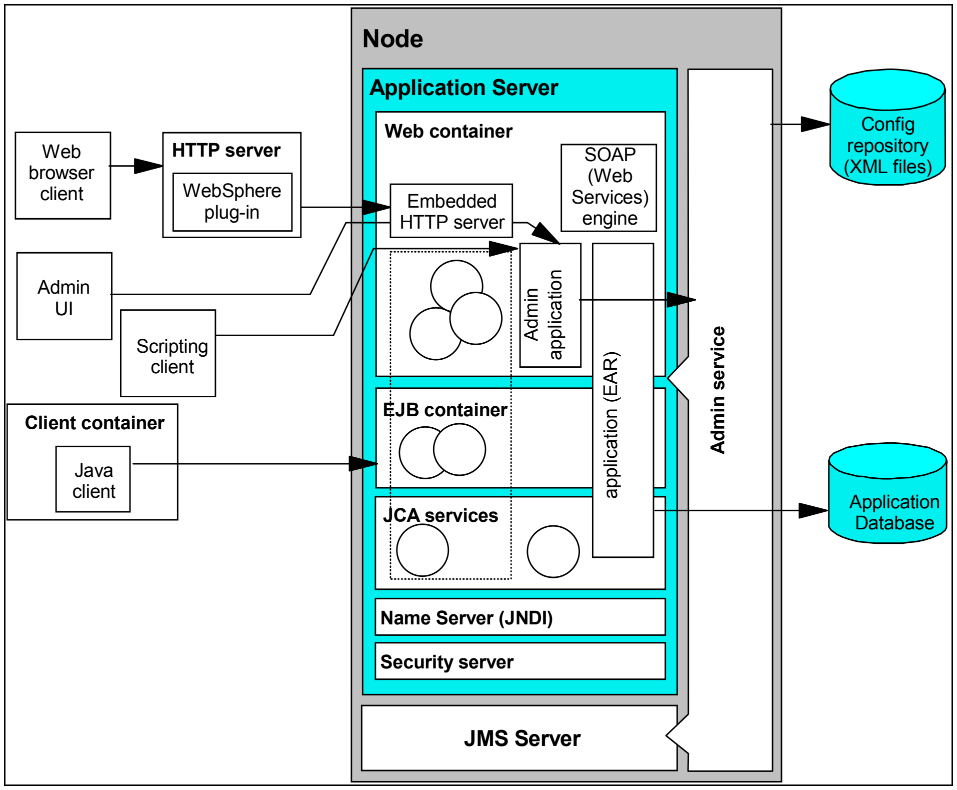


Figure 2-2 WebSphere Application Server components

2.1.3 Network Deployment configuration

The Network Deployment configuration offers central administration and workload management. A Network Deployment environment consists of one or more Base installations and a Deployment Manager installation. The Base application servers are added to the cell and managed by the Deployment Manager. The Network Deployment package also includes the Web Services Private UDDI Registry and Web Services Gateway.

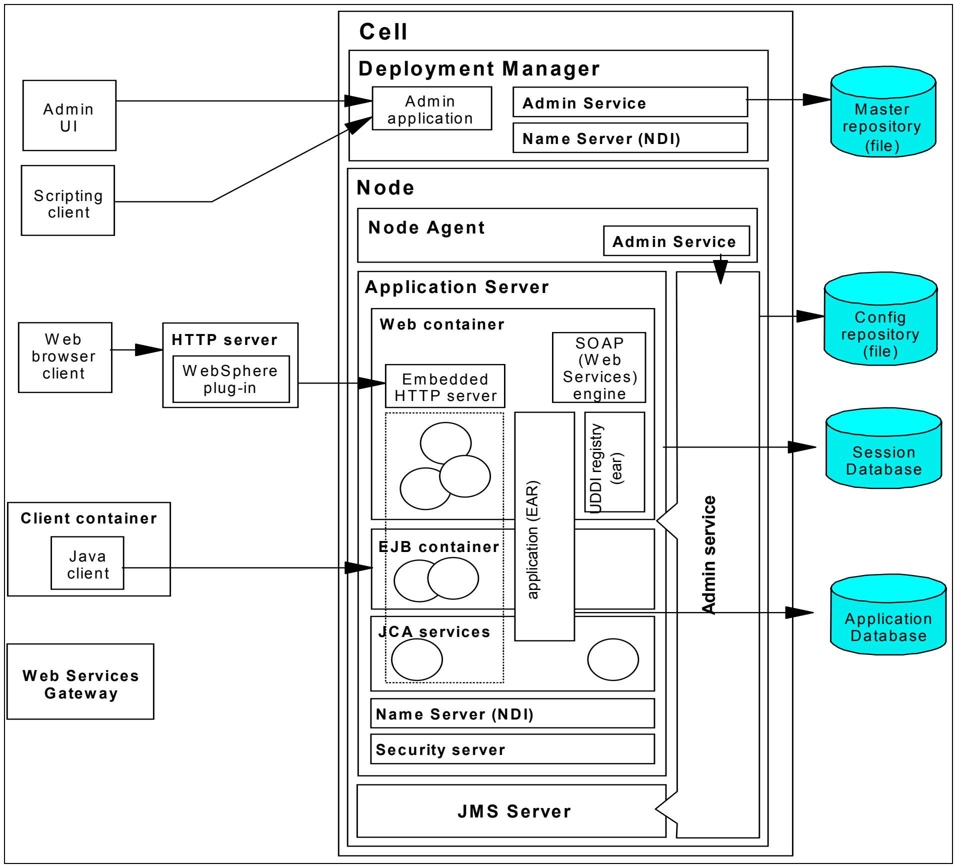


Figure 2-3 WebSphere Application Server Network Deployment components

2.1.4 Enterprise configuration

Like the Network Deployment package, the Enterprise package contains the base WebSphere Application Server and the Deployment Manager. In addition, it adds several programming model extensions (PME) to the application server. These extensions are delivered in different forms, including services, APIs, wizards for development, and deployment extensions. To support these extensions, the Enterprise package adds a business process container to the application server and the necessary administrative console features to support the runtime environment.

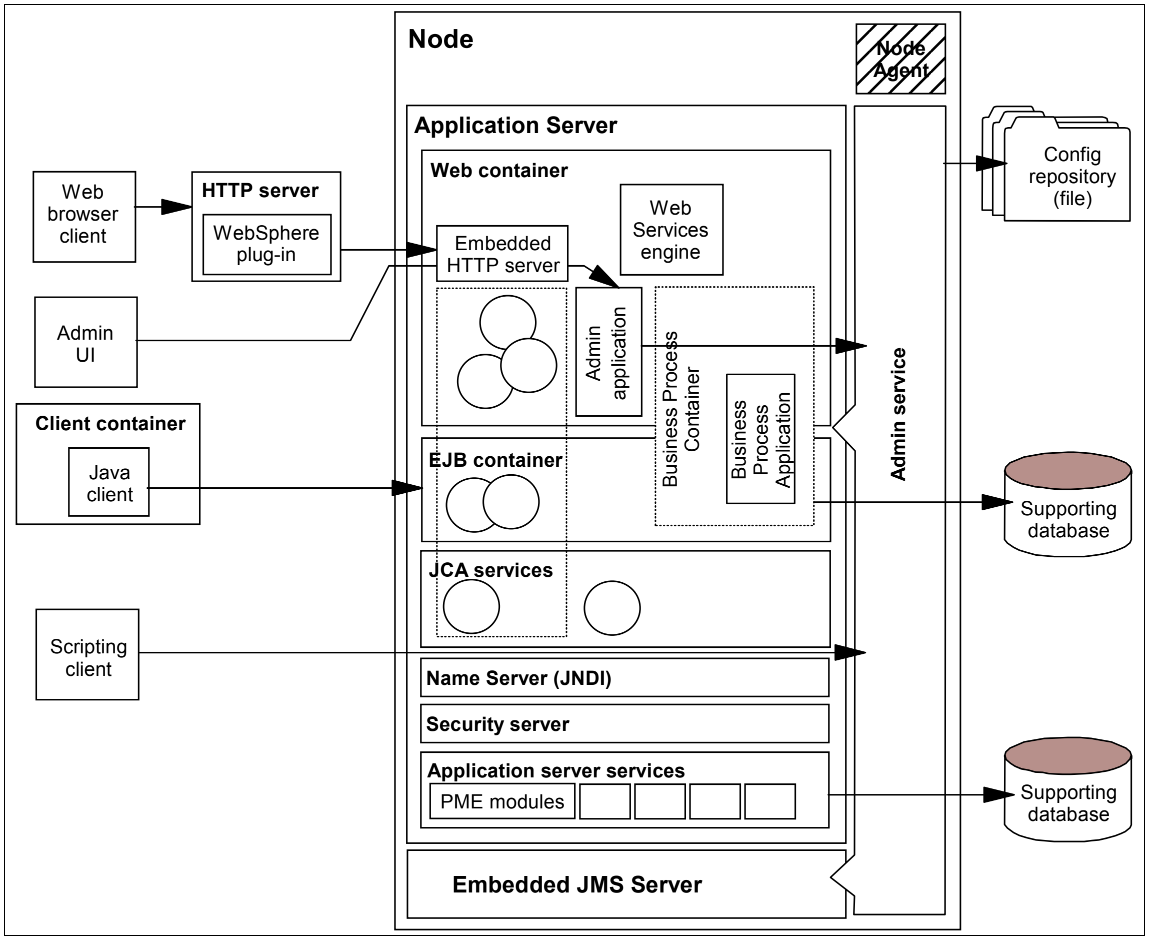


Figure 2-4 WebSphere Application Server Enterprise architecture

2.2 Cells, nodes and servers

Regardless of the configuration, the WebSphere Application Server is organized based on the concept of cells, nodes and servers. While all these elements are present in each configuration, cells and nodes do not play an important role until you reach the level of the Network Deployment and Enterprise configurations.

Servers

Servers perform the actual code execution. There are several types of servers, depending on the configuration. Each server runs in its own JVM.

- Application servers:

  The application server is the primary runtime component in all configurations. This is where the application actually executes. All WebSphere Application Server configurations can have one or more application servers. In the Express and Base configurations, each application server functions as a separate entity. There is no workload distribution or common administration among application servers.

  In the Network Deployment configuration, multiple application servers are maintained from a central administration point. In addition, application servers can be clustered for workload distribution.

- JMS servers:

  The Base, Network Deployment, and Enterprise configurations provide an embedded JMS server for messaging support. In the Base configuration, the JMS server functions are integrated into the application server. In the Network Deployment and Enterprise, the JMS server runs in a separate JVM. There is one JMS server per node.

Nodes and node agents
A node is a logical grouping of WebSphere-managed server processes that share common configuration and operational control. A node is generally associated with one physical installation of WebSphere Application Server. In the Express and Base configurations of WebSphere Application Server, there is only one node.

As you move up through the more advanced WebSphere Application Server configurations, the concepts of configuring multiple nodes from one common administration server and workload distribution among nodes are introduced. In these centralized management configurations, each node has a node agent that works with a Deployment Manager to manage administration processes.

Cells
A cell is a grouping of nodes into a single administrative domain. In the Base and Express configurations, a cell contains one node. That node may have multiple servers, but the configuration files for each server are stored and maintained individually.

With the Network Deployment and Enterprise configurations, a cell can consist of multiple nodes, all administered from a single point. The configuration and application files for all nodes in the cell are centralized into a cell master configuration repository. This centralized repository is managed by the Deployment Manager process and synchronized out to local copies held on each of the nodes.

2.3 Servers

WebSphere Application Server uses servers to provide the functions required to host applications. Application servers execute user applications. They may act as individual servers or as a member of a cluster. JMS servers provide the embedded messaging support.

Table 2-1 WebSphere Application Server server support

	Express	Base	ND	Enterprise
Application server	Yes	Yes	Yes	Yes
JMS server	No	Yes (integrated into application server)	Yes	Yes
Application server clustering	No	No	Yes	Yes

2.3.1 Application server

Application servers provide the runtime environment for application code. They provide containers and services that specialize in enabling the execution of specific Java application components.

Each application server runs in its own Java virtual machine (JVM). In all configurations, one application server, called server1, is available automatically after installation. Additional application servers can be created, but unless you are using the Network Deployment or Enterprise configuration, the application servers are configured independently and have no workload distribution capabilities.

2.3.2 JMS server

Messaging support is provided through the use of one of the following JMS providers:

- WebSphere JMS provider (embedded)
- WebSphere MQ JMS provider
- Generic JMS provider

If you use the WebSphere MQ JMS provider or a generic JMS provider, the JMS functions are provided by an external product.

The embedded WebSphere JMS provider, on the other hand, is included with all WebSphere Application Server configurations except the Express configuration. The integrated messaging functions provided by the WebSphere JMS provider include support for message-driven beans, point-to-point and publish/subscribe styles of messaging, and integration with the transaction management service.

The functions of the embedded WebSphere JMS provider are implemented by a JMS server. In the Base configuration, the JMS server runs in the same JVM as the application server. From a configuration standpoint, it is a part of the application server.

In the Network Deployment and Enterprise configurations, the JMS server is separated from the application server and runs in a separate, dedicated JVM. In this case, the configuration for the JMS server is independent of application servers.

2.3.3 Clusters

The Network Deployment and Enterprise configurations offer the option of using application server clustering to provide enhanced workload distribution. A *cluster* is a logical collection of application server processes, with the sole purpose of providing workload balancing.

Application servers that belong to a cluster are "members" of that cluster and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data.

For example, one cluster member might be running on a large multi-processor server while another member of that same cluster might be running on a small laptop. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of the configuration, they are identical.

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster), or on a combination of the two.

Application management

When you install, update, or delete an application, the updates are automatically distributed to all members in the cluster.

Workload management

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS.

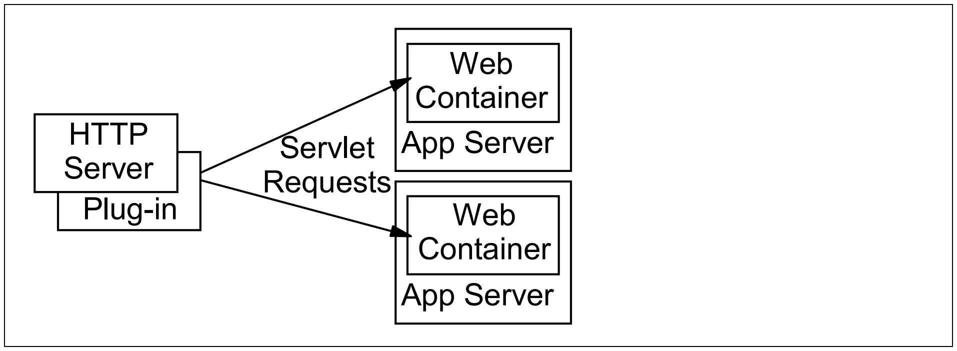


Figure 2-5   Plug-in (Web container) workload management

This routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in will send equal requests to all members of the cluster, assuming no strong affinity configurations. If the weights are scaled in a range of zero to twenty, the plug-in will more often route requests to those cluster members with the higher weight value. A rule of thumb formula for determining routing preference would be:

% routed to Server1 = weight1 / (weight1 + weight2 + ... + weight*n*)

where there are n cluster members in the cluster.

The Web server plug-in will temporarily route around unavailable cluster members.

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers with the EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers.

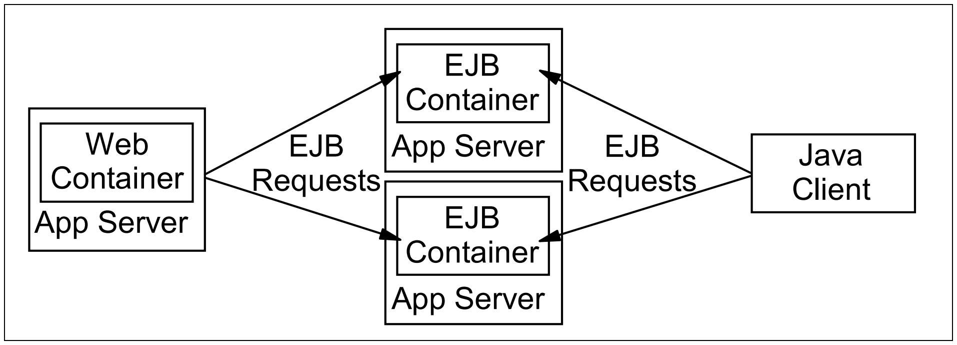


Figure 2-6 EJB workload management

In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

The server weighted round robin routing policy will ensure a distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster would be that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism will send more requests to the higher weight value servers than to the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members. You can also choose to have requests sent to the node on which the client resides as the preferred routing method. In this case, only cluster members on that node will be chosen (using the round robin weight method). Cluster members on remote nodes will only be chosen if a local server is not available.

2.4 Containers

The J2EE 1.3 specification defines the concept of containers to provide runtime support for applications. There are two containers in the application server implementation:

- Web container - processes HTTP requests, servlets and JSPs.
- EJB container - processes enterprise beans (EJBs).

In addition, there is an application client container which can run on the client machine.

Table 2-2 WebSphere Application Server container support

	Express	Base	ND	Enterprise
Web container	Yes	Yes	Yes	Yes
EJB container	No	Yes	Yes	Yes
Application client container	No	Yes	Yes	Yes

2.4.1 Web container

The Web container processes servlets, JSP files and other types of server-side includes. Each application server runtime has one logical Web container, which can be modified, but not created or removed.

- Servlet processing: when handling servlets, the Web container creates a request object and a response object, then invokes the servlet service method. The Web container invokes the servlet's destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.
- Embedded HTTP server: the Web container runs an embedded HTTP server for handling HTTP(S) requests from external Web server plug-ins or Web browsers. The embedded Web server is based on the IBM HTTP Server product.

  Directing client requests to the embedded Web server is useful for testing or development purposes and, in the Express configuration, can be considered for production use. In the more advanced configurations, the use of an external Web server and Web server plug-in as a front end to the Web container is more appropriate for a production environment.
- Session management: support is provided for the javax.servlet.http.HttpSession interface described in the Servlet API specification.
- Web services engine: Web services are provided as a set of APIs in cooperation with the J2EE applications. Web services engines are provided to support SOAP.

Web server plug-ins

Although the Web container has an embedded HTTP server, a more likely scenario is that an external Web server will be used to receive client requests. The Web server can serve requests that do not require any dynamic content, for example, HTML pages. However, when a request requires dynamic content

(JSP/servlet processing), it must be forwarded to WebSphere Application Server for handling.

The mechanism to accomplish this is provided in the form of a Web server plug-in. The plug-in is included with the WebSphere Application Server package for installation on a Web server. An XML configuration file, configured on the WebSphere Application Server, is copied to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When a request for an application server is received, it is forwarded to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPs to transmit the request.

2.4.2 EJB container

The EJB container provides all the runtime services needed to deploy and manage enterprise beans. It is a server process that handles requests for both session and entity beans.

The enterprise beans (packaged in EJB modules) installed in an application server do not communicate directly with the server; instead, the EJB container provides an interface between the EJBs and the server. Together, the container and the server provide the bean runtime environment.

The container provides many low-level services, including threading and transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained beans. A single container can host more than one EJB JAR file.

2.4.3 Client application container

The client application container is a separately installed component on the client's machine. It allows the client to run applications in an EJB-compatible J2EE environment.

There is a command-line executable (**`launchClient`**) which is used to launch the client application along with its client container runtime.

2.5 Application server services

The application server provides other services besides the containers.

Table 2-3 WebSphere Application Server services

	Express	**Base**	**ND**	**Enterprise**
JCA services	Yes[1]	Yes	Yes	Yes
Transaction service	No	Yes	Yes	Yes
Dynamic cache service	No	Yes	Yes	Yes
Message listener service	No	Yes	Yes	Yes
ORB service	No	Yes	Yes	Yes
Admin service (JMX)	Yes	Yes	Yes	Yes
Diagnostic trace service	Yes	Yes	Yes	Yes
Debugging service	Yes	Yes	Yes	Yes
Name service (JNDI)	Yes	Yes	Yes	Yes
Performance Monitoring Interface (PMI) service	No	Yes	Yes	Yes
Security service (JAAS and Java 2 security)	JAAS but not Java 2	Yes	Yes	Yes
Business process engine	No	No	No	Yes

1. Included to provide JDBC support, but not configurable.

2.5.1 JCA services

Connection management for access to enterprise information systems (EIS) in WebSphere Application Server is based on the J2EE Connector Architecture (JCA) specification, also sometimes referred to as J2C. The connection between the enterprise application and the EIS is done through the use of EIS-provided resource adapters, which are plugged into the application server. The architecture specifies the connection management, transaction management and security contracts that exist between the application server and EIS.

The Connection Manager in the application server pools and manages connections. It is capable of managing connections obtained through both resource adapters defined by the JCA specification and data sources defined by the JDBC 2.0 Extensions Specification.

Note that in the Express configuration, adding resource adapters is not supported. However, the JCA support is included to handle connectivity to JDBC resources. The connectivity is handled through a resource adapter provided by WebSphere Application Server.

2.5.2 Transaction service

WebSphere applications can use transactions to coordinate multiple updates to resources as one unit of work such that all or none of the updates are made permanent. Transactions are started and ended by applications or the container in which the applications are deployed.

WebSphere Application Server is a transaction manager that supports the coordination of resource managers through their XAResource interface and participates in distributed global transactions with other OTS 1.2 compliant transaction managers (for example J2EE 1.3 application servers).

WebSphere applications can also be configured to interact with databases, JMS queues, and JCA connectors through their local transaction support when distributed transaction coordination is not required.

The way that applications use transactions depends on the type of application component, as follows:

- A session bean can either use container-managed transactions (where the bean delegates management of transactions to the container) or bean-managed transactions (where the bean manages transactions itself).
- Entity beans use container-managed transactions.
- Web components (servlets) use bean-managed transactions.

In WebSphere Application Server, transactions are handled by three main components:

- A transaction manager which supports the enlistment of recoverable XAResources and ensures that each such resource is driven to a consistent outcome either at the end of a transaction or after a failure and restart of the application server.
- A container in which the J2EE application runs. The container manages the enlistment of XAResources on behalf of the application when the application performs updates to transactional resource managers (for example, databases). Optionally, the container can control the demarcation of transactions for enterprise beans configured for container-managed transactions.
- An application programming interface (UserTransaction) which is available to bean-managed enterprise beans and servlets. This allows such application components to control the demarcation of their own transactions.

2.5.3 Dynamic caching

The dynamic cache service improves performance by caching the output of servlets, commands and JSP files. The dynamic cache works within an application server, intercepting calls to cacheable objects, for example through a servlet's service() method or a command's execute() method, and either stores the object's output to or serves the object's content from the dynamic cache.

Because J2EE applications have high read-write ratios and can tolerate small degrees of latency in the currency of their data, the dynamic cache can create an opportunity for significant gains in server response time, throughput, and scalability.

The following caching features are available in WebSphere Application Server.

- Cache replication:

  Cache replication among cluster members takes place using the WebSphere internal replication service. Data is generated one time and copied or replicated to other servers in the cluster, thus saving execution time and resources.
- Cache disk offload:

  By default, when the number of cache entries reaches the configured limit for a given WebSphere server, eviction of cache entries takes place, allowing new entries to enter the cache service. The dynamic cache includes an alternative feature named *disk offload*, which copies the evicted cache entries to disk for potential future access.

- Edge Side Include caching:

  The Web server plug-in contains a built-in ESI processor. The ESI processor has the ability to cache whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an in-memory cache, not a disk cache; therefore, the cache entries are not saved when the Web server is restarted.

- External caching:

  The dynamic cache has the ability to control caches outside of the application server, such as IBM Edge Server, a non-z/OS IBM HTTP Server's FRCA cache, and a non-z/OS WebSphere HTTP Server plug-in ESI Fragment Processor. When external cache groups are defined, the dynamic cache matches externally cacheable cache entries with those groups, and pushes cache entries and invalidations out to them. This allows WebSphere to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving savings in performance.

2.5.4 Message listener service

The message listener service is an extension to the JMS functions of the JMS provider. It provides a listener manager that controls and monitors one or more JMS listeners. Each listener monitors a JMS destination on behalf of a deployed message-driven bean.

2.5.5 Object Request Broker service

An Object Request Broker (ORB) manages the interaction between clients and servers, using IIOP. It enables clients to make requests and receive responses from servers in a network-distributed environment.

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as if the remote objects were located in the same running process as the client, providing location transparency. The client calls an operation on a local object, known as a stub. Then the stub forwards the request to the desired remote object, where the operation is run and the results are returned to the client.

The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request on the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB demarshals the returned results and passes the result to the stub, which, in turn, returns to the client application, as if the operation had been run locally.

WebSphere Application Server uses an ORB to manage communication between client applications and server applications, as well as communication among product components.

2.5.6 Name service

Each application server hosts a name service that provides a Java Naming and Directory Interface (JNDI) name space. The service is used to register resources hosted by the application server. The JNDI implementation in WebSphere Application Server is built on top of a Common Object Request Broker Architecture (CORBA) naming service (CosNaming).

JNDI provides the client-side access to naming and presents the programming model used by application developers. CosNaming provides the server-side implementation and is where the name space is actually stored. JNDI essentially provides a client-side wrapper of the name space stored in CosNaming, and interacts with the CosNaming server on behalf of the client.

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications. These objects are bound into a mostly hierarchical structure, referred to as a *name space*. The name space structure consists of a set of name bindings, each consisting of a name relative to a specific context and the object bound with that name. The name space can be accessed and manipulated through a name server.

The following are features of a WebSphere Application Server name space:

- The name space is distributed

  For additional scalability, the name space for a cell is distributed among the various servers. The Deployment Manager, node agent and application server processes all host a name server. In previous releases, there was only a single name server for an entire administrative domain.

  In WebSphere releases prior to V5.0, all servers shared the same default initial context, and everything was bound relative to that initial context. In WebSphere Application Server V5, the default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server with which they are associated.

- Transient and persistent partitions

  The name space is partitioned into transient areas and persistent areas. Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell persistent root, which can be used for cell-scoped persistent bindings, and a node persistent root, which can be used to bind objects with a node scope.

- Federated name space structure

  A name space is a collection of all names bound to a particular name server. A name space can contain naming context bindings to contexts located in other servers. If this is the case, the name space is said to be a *federated name space*, because it is a collection of name spaces from multiple servers. The name spaces link together to cooperatively form a single logical name space.

  In a federated name space, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

  In the Network Deployment and Enterprise configurations, the name space for the cell is federated among the Deployment Manager, node agents, and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell name space, with the various server roots and persistent partitions of the name space being interconnected by means of the single logical name space.

- Configured bindings

  The configuration graphical interface and script interfaces can be used to configure bindings in various root contexts within the name space. These bindings are read-only and are bound by the system at server startup.

- Support for CORBA Interoperable Naming Service (INS) object URLs

  WebSphere Application Server V5 contains support for CORBA object URLs (corbaloc and corbaname) as JNDI provider URLs and lookup names, as required by the J2EE 1.3 specification.

Figure 2-7 on page 47 summarizes the naming architecture and its components.

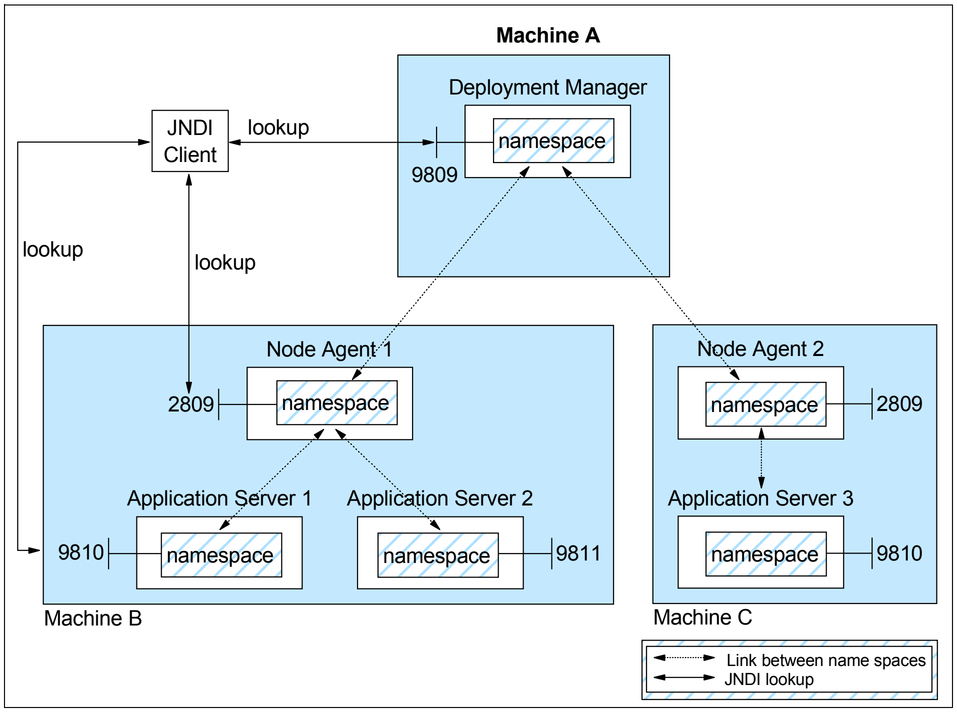


Figure 2-7 Naming topology

2.5.7 PMI service

WebSphere Application Server collects data on runtime and applications through the Performance Monitoring Infrastructure (PMI). This infrastructure is compatible with and extends the JSR-077 specification.

PMI uses a client-server architecture. The server collects performance data from various WebSphere Application Server components and stores it in memory. This data consists of counters such as servlet response time and data connection pool usage. The data can then be retrieved using a Web client, Java client or JMX client. WebSphere Application Server contains Tivoli Performance Viewer, a Java client which displays and monitors performance data.

WebSphere Application Server also collects data by timing requests as they travel through the product components. PMI request metrics log time spent in major components, such as Web containers, EJB containers, and databases. These data points are recorded in logs and can be written to Application Response Time (ARM) agents used by Tivoli monitoring tools.

The PMI services can be enabled for an application server or node agent using the administrative console. The type and level of data collected can be controlled through the administrative console (or wsadmin) or from the Tivoli Performance Viewer.

2.5.8 Security service

Each application server JVM hosts a security service that uses the security settings held in the configuration repository to provide authentication and authorization functionality.

2.5.9 Admin service

The admin service runs within each server JVM. In the Base configuration, the admin service runs in the application server. In the Network Deployment and Enterprise configurations, each of the following hosts an admin service:

- Deployment Manager
- Node agent
- Application server
- JMS server

The admin service provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository in the server's file system.

The admin service has a security control and filtering functionality, providing different levels of administration to certain users or groups using the following admin roles:

- Administrator
- Monitor
- Configurator
- Operator

2.5.10 Enterprise services

The WebSphere Application Server architecture is designed in such a way that new application server services can be easily developed and plugged into the existing server architecture. The Programming Model Extensions are implementations of service extensions installed on top of the base application server.

The question arises: how can one use the services provided in WebSphere Application Server Enterprise? Think about the base services provided by the application server, such as the Web container, EJB container, and embedded

HTTP Server. These services are either accessed by using a specific API, for example servlet API, EJB API, or the service interacts directly with the deployed code and content, for example serving HTML files.

The services in WebSphere Application Server Enterprise work in the same way. Figure 2-8 summarizes how an enterprise application can access the enterprise services.

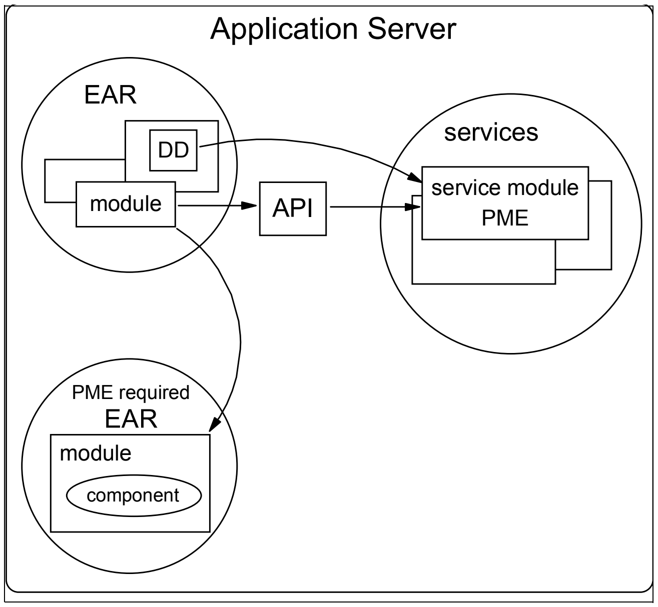


Figure 2-8 Using the application server services

Some services can be accessed using the provided API, for example the Scheduler service, Internationalization (I18N) service, or Shared Work Area service. Services such as last participant support work behind the scenes and use the deployment descriptors or extensions in the descriptors. Other services, such as dynamic query, require a specific enterprise application that provides access to the services to other applications.

The following table shows a list of the extensions and the provided method of access to them.

Table 2-4   *Accessing PMEs in WebSphere Enterprise*

Extension	**API access**	**Deployment Descriptor**	**Special EAR provided**
Process Choreographer	X (for the client)		X (application itself is a special EAR)
Extended messaging	X (Wizards provided in Studio)		
Asynchronous beans	X		
Application profiling, access intent		X	
Business rule beans	X		X
Dynamic query	X		X
Startup beans	X		
Scheduler service	X		
ActivitySession	X		
Last participant support		X	
Object pools	X		
Shared work area	X		
Internationalization (I18N) service	X		
CORBA support	X		

2.6 Business process engine

Table 2-5   *WebSphere Application Server business process engine support*

	Express	**Base**	**ND**	**Enterprise**
Business process engine	No	No	No	Yes

The Enterprise configuration uses an engine called the *business process engine* to support the processing of short and long running business processes. This engine is also sometimes referred to as the *business process container* although it is really a special enterprise application and not an application server container at all. The administration process hides these details from the user, so it really looks and works like an individual container.

The business process applications, developed using the Integrated Edition of WebSphere Studio, are also deployed as enterprise applications. The enterprise administrative processes ensure that these applications are deployed into the business process container.

The business process container application is the first enterprise application loaded when the application server is started. Its classloader is aware of other enterprise application modules and loads the business process applications before other applications are loaded.

Business process applications usually have three modules:

- A process EJB module that implements the process flow.
- A process Web client that provides the user interface for the process.
- A flow archive (.far) file that contains the flow descriptions implemented in the application. These flow descriptions are stored in .fdml format. This module is automatically generated by WebSphere Studio and stored under the root EAR module as a packaged file.

The business process engine requires a database to persist the process templates, instances and states. The database is registered in WebSphere as a data source and can be any database type supported by WebSphere.

Multiple instances of the business process container can run in parallel on a single node or distributed in a cluster. Both the clustering capabilities of WebSphere Application Server for IIOP requests and the clustering capabilities of WebSphere MQ for JMS-based requests can be exploited. The business process database acts as a central state repository.

2.7 Virtual hosts

A virtual host is a configuration enabling a single host machine to resemble multiple host machines. It allows a single physical machine to support several independently configured and administered applications. It is not associated with a particular node. It is a configuration, rather than a “live object,” which is why it can be created, but not started or stopped.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example yourHostName:80. When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an HTTP 404 error is returned to the browser.

WebSphere Application Server provides two default virtual hosts:

- **default_host**

  Used for accessing most applications. For all configurations except Express, the default settings for default_host map to all requests for any alias on ports 80, 9443, and 9080. In the Express configuration, the default_host port settings are 80, 7443, and 7080.

  For example:

  ```
  http://localhost:80/servlet/snoop
  http://localhost:9080/servlet/snoop
  ```

- **admin_host**

  Specifically configured for accessing the WebSphere Application Server administrative console. Other applications are not accessible through this virtual host. For all configurations except Express, the default settings for admin_host map to requests on ports 9090 and 9043. In the Express configuration, the default_host port settings are 7090 and 7043.

  For example:

  ```
  http://localhost:9090/admin
  ```

2.8 Session management

In many Web applications, users dynamically collect data as they move through the site based on a series of selections on pages they visit. Where the user goes next, and what the application displays as the user's next page (or next choice) may depend on what the user has chosen previously from the site. In order to maintain this data, the application stores it in a “session.”

WebSphere supports three approaches to track sessions:

- SSL session identifiers - SSL session information is used to track the HTTP session ID.
- Cookies - The application server session support generates a unique session ID for each user, and returns this ID to the user’s browser using a cookie. The

default name for the session management cookie is JSESSIONID. Using cookies is the most common method of session management.

- URL rewriting

Session data can be kept in local memory cache, stored externally on a database, or kept in memory and replicated among application servers.

Table 2-6 shows the session support for each WebSphere Application Server configuration.

Table 2-6 WebSphere Application Server session management support

	Express	**Base**	**ND**	**Enterprise**
Cookies	Yes	Yes	Yes	Yes
URL rewriting	Yes	Yes	Yes	Yes
SSL session identifiers	Yes	Yes	Yes	Yes
In memory cache	Yes	Yes	Yes	Yes
Session persistence using a database	No	Yes	Yes	Yes
Memory-to-memory session persistence	No	No	Yes	Yes

The Servlet 2.3 specification defines session scope at the Web application level, meaning that session information can only be accessed by a single Web application. However, there may be times when there is a logical reason for multiple Web applications to share information, for example a user name. WebSphere V5.0 provides an IBM extension to the specification allowing session information to be shared among Web applications within an enterprise application. This option is offered as an extension to the application deployment descriptor. No code change is necessary to enable this option, which is specified during application assembling.

2.8.1 Session persistence

Many Web applications use the simplest form of session management, the in-memory local session cache. The local session cache keeps session information in memory and local to the machine and WebSphere Application Server where the session information was first created. Local session management does not share user session information with other clustered machines. Users only obtain their session information if they return to the machine and WebSphere Application Server holds their session information on subsequent accesses to the Web site.

Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server, but also destroys any sessions managed by those instances.

By default, WebSphere places session objects in memory. However, with the Base, Network Deployment, and Enterprise configurations, the administrator has the option of enabling persistent session management. This instructs WebSphere to place session objects in a persistent store. Using a persistent store allows the user session data to be recovered by another cluster member after a cluster member in a cluster fails or is shut down. Two options for session persistence are available:

- Database

  In a multi-server environment, session information can be stored in a central session database for session persistence. The multiple application servers hosting a particular application need to share this database information in order to maintain session states for the stateful components.

- Memory-to-memory using WebSphere internal messaging

  WebSphere internal messaging enables sessions to be shared among application servers without using a database. Using this method, sessions are stored in the memory of an application server, providing the same functionality as a database for session persistence. Two topology options are offered:

  – Peer-to-peer topology: each application server stores sessions in its own memory. It also stores sessions to and retrieves sessions from other application servers. In other words, each application server acts as a client by retrieving sessions from other application servers, and each application server acts as a server by providing sessions to other application servers.

  – Client/server topology: application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that only store sessions but do not respond to the users' requests. Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact.

The internal messaging system is not associated with the embedded WebSphere JMS provider or JMS server. The necessary code is provided as part of the core WebSphere Application Server libraries.

2.9 Web services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. WebSphere Application Server can act as both a Web service provider and as a requester. As a provider, it hosts Web services that are published for use by clients. As a requester, it hosts applications that invoke Web services from other locations.

WebSphere Application Server supports SOAP-based Web service hosting and invocation.

Table 2-7 WebSphere Application Server server support

	Express	Base	ND	Enterprise
Base Web services support	Yes	Yes	Yes	Yes
Private UDDI Registry	No	No	Yes	Yes
Web Services Gateway	No	No	Yes	Yes
Enterprise Web services	Yes	Yes	Yes	Yes
JCA Web services	No	No	No	Yes
Web services internationalization	No	No	No	Yes
[1] Not yet supported in V5.0				

The Web services support includes the following:

- Web Services Description Language (WSDL), an XML-based description language that provides a way to catalog and describe services. It describes the interface of Web services (parameters and result), the binding (SOAP, EJB), and the implementation location.

- Universal Discovery Description and Integration (UDDI), a global, platform-independent, open framework to enable businesses to discover each other, define their interaction, and share information in a global registry.
- Simple Object Access Protocol (SOAP), a lightweight protocol for exchange of information in a decentralized, distributed environment.
- eXtensible Markup Language (XML), which provides a common language for exchanging information.
- Web Services Invocation Framework (WSIF), a runtime architecture for service-oriented applications that provides a binding-independent mechanism for Web service invocation. WSIF enables easy encapsulation of back-end systems connected through the J2EE Connector Architecture (JCA) as Web services.
- JAX-RPC (JSR 101) provides the core programming model and bindings for developing and deploying Web services on the Java platform. It is a Java API for XML-based RPC and supports only JavaBeans as a Web service provider.
- Enterprise Web services (JSR 109): this adds EJBs and XML deployment descriptors to JSR 101.
- WS-Security: this specification covers a standard set of SOAP extensions that can be used when building secure Web services to provide integrity and confidentiality. It is designed to be open to other security models including PKI, Kerberos, and SSL. WS-Security provides support for multiple security tokens, multiple signature formats, multiple trust domains, and multiple encryption technologies. It includes security token propagation, message integrity, and message confidentiality. The specification is proposed by IBM, Microsoft, and VeriSign for review and evaluation. In the future, it will replace existing Web services security specifications from IBM and Microsoft including SOAP Security Extensions (SOAP-SEC), Microsoft's WS-Security and WS-License, and IBM's security token and encryption documents.

Enterprise services (JCA Web services)

Enterprise services offer access over the Internet to applications in a platform-neutral and language-neutral fashion. They offer access to enterprise information systems (EIS) and message queues and can be used in a client/server configuration without the Internet. Enterprise services can access applications and data on a variety of platforms and in a variety of formats.

An enterprise service wraps a software component in a common services interface. The software component is typically a Java class, EJB, or JCA resource adapter for an EIS. In services terminology, this software component is known as the *implementation*. Enterprise services primarily use WSDL and WSIF to expose an implementation as a service.

Using the Integrated Edition of WebSphere Studio, you can turn CICS® and IMS™ transactions using Java Connector Architecture (JCA) into Web services.

Web service client (requester)

Applications that invoke Web services are known as Web service clients or Web service requestors. An application that acts as a Web service client is deployed to WebSphere Application Server like any other enterprise application. No additional configuration or software is needed for the Web services client to function. Web services clients can also be stand-alone applications.

A Web service client will bind to a Web service server to invoke the Web service. The binding is done using a service proxy (or *stub*), which is generated based on the WSDL document for the Web service. This service proxy contains all the needed information to invoke the Web service and is used locally by the clients to access the business service. The binding can also be done dynamically using WSIF.

Web service provider

An application that acts as a Web service is deployed to WebSphere Application Server like any other enterprise application. The Web services are contained in Web modules or EJB modules.

Publishing the Web service to a UDDI registry makes it available to anyone searching for it. Web services can be published to a UDDI registry using the Web Services Explorer provided with WebSphere Studio Application Developer, Integration Edition, and Enterprise Developer configurations.

When using WebSphere Studio V5 to package the application for deployment, no additional configuration or software is needed for the Web services client to function. The SOAP servlets are automatically added and a SOAP admin tool is included in a Web module.

If not, you will need to use soapearenabler.bat, found in the WebSphere bin directory, to enable the SOAP services within the EAR file and add the SOAP admin tool.

Enterprise Web Services

The Enterprise Web Services, based on the JSR 109 specification request, provide the use of JAX-RPC in a J2EE environment defining the runtime architecture, as well as the implementation and deployment of Web services in a generic J2EE server. The specification defines the programming model and architecture for implementing Web services in Java based on JSRs 67, 93, 101, and future JSRs related to Web services standards. The list of JSRs may be found at:

http://www.jcp.org/en/jsr/all

2.9.1 IBM WebSphere UDDI Registry

WebSphere Application Server V5 provides a private UDDI registry that implements V2.0 of the UDDI specification. This enables the enterprise to run its own Web services broker within the company or provide brokering services to the outside world.

The UDDI registry is installed as a Web application by the administrator on one application server (or cluster). A DB2 database is used to store registry data. In a test environment, Cloudscape™ can be used.

There are three interfaces to the registry for inquiry and publish:

- Through the UDDI SOAP API.
- Through the UDDI EJB client interface.
- Through the UDDI user console. This Web-based GUI interface can be used to publish and to inquire about UDDI entities but only provides a subset of the UDDI API functions.

Security for the UDDI registry is handled using WebSphere security. To support the use of secure access with the IBM WebSphere UDDI Registry, you need to configure WebSphere to use HTTPS and SSL.

2.9.2 Web Services Gateway

The Web Services Gateway bridges the gap between Internet and intranet environments during Web service invocations. The Gateway builds upon the Web services Definition Language (WSDL) and the Web Services Invocation Framework (WSIF) for deployment and invocation.

The primary function of the Web Services Gateway is to map an existing WSDL-defined Web service to a new service that is offered by the Gateway to others. The Gateway thus acts as a proxy. External services are imported into the Gateway and made available to the enterprise as internal proxy services. Likewise, internal services are imported into the Gateway and made available as external proxy services. These services are also published to the relevant UDDI registries where required.

The Web Services Gateway is installed as a Web application by the administrator.

Exposing Web services to the outside world

To expose an internal service for outside consumption, the Gateway takes the WSDL file for the existing service and generates a new WSDL file that can be shared with outside requestors. The interface described in the WSDL is exactly the same, but the service endpoint is changed to the Gateway, which is now the official endpoint for the service client.

You might also consider applying special security constraints to restrict the access to either specific users and/or specific methods of your service. This scenario is illustrated in Figure 2-9.

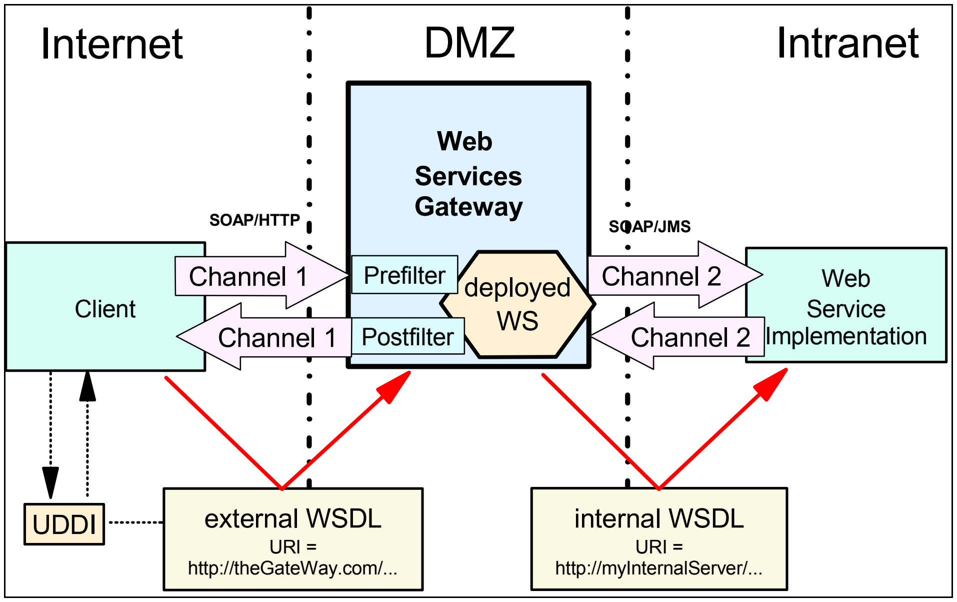


Figure 2-9 Exposing Web services through the Gateway

Importing Web services

An external service may be imported and made available as an internal service. This will help the internal service requestors invoke the service as if it were running on the Gateway. Again, a new WSDL is generated by the Gateway showing the same interface but naming the Gateway as a service provider rather than the real internal server. The Gateway then reroutes all requests to the actual implementation specified in the original WSDL.

Of course, every client could access external Web services by traditional means, but if you add the Gateway as an additional layer in between, clients do not have to change anything if the service implementor changes. This scenario is very similar to the illustration in Figure 2-9, with the difference that the client resides in the intranet and the Web service implementation is located at a site on the Internet.

Managing channels

A request for a service may originate in one protocol, but the service may be invoked in some other protocol by using the transformation function. Let us consider the case that an internal service is available on SOAP over JMS, but should be invoked using SOAP over HTTP. To handle such cases, you need to define a set of channels corresponding to the specific protocols on which you wish the Gateway to receive messages. The outbound channels over which the Gateway invokes the target Web services are defined implicitly by the bindings in the WSDL imported in the Gateway and do not have to be explicitly configured. By using a different inbound channel to those defined in the imported WSDL bindings, you can switch the protocol.

Managing filters

You can develop and use one or more filters for your Web services, which means that some specific task is done using the input and/or output messages to and from your Web service. There are standard filters as well as the possibility of creating your own.

UDDI publication and lookup

The Gateway facilitates working with UDDI registries. As you map a service for external consumption using the Gateway, you can publish the exported WSDL in the UDDI registry. When the services in the Gateway are modified, the UDDI registry is updated with the latest changes.

2.10 Security

WebSphere Application Server security sits on top of the operating system security and the security features provided by other components, including the Java language.

- Operating system security protects sensitive WebSphere configuration files and authenticates users when the operating system user registry is used for authentication.
- Standard Java security is provided through the Java Virtual Machine (JVM) used by WebSphere and the Java security classes.
- The Java 2 Security API provides a means to enforce access control, based on the location of the code and who signed it. Java 2 security guards access to system resources such as file I/O, sockets, and properties. WebSphere global security settings allow you to enable or disable Java 2 security and

provide a default set of policies. Java 2 security can be activated or inactivated independently from WebSphere global security; however, when global security is enabled, by default Java 2 security is also enabled.

The current principal of the thread of execution is not considered in the Java 2 security authorization. There are instances where it is useful for the authorization to be based on the principal, rather than the code base and the signer. Java Authentication and Authorization Services (JAAS) is a standard Java API that allows the Java 2 security authorization to be extended to the code based on the principal as well as the code base and signers.

JAAS is a standard extension to the Java 2 SDK v1.3 and is part of the Java 2 SDK V1.4. The JAAS programming model allows the developer to design application authentication in a pluggable fashion, which makes the application independent of the underlying authentication technology. JAAS does not require Java 2 security to be enabled.

- The Common Secure Interoperability protocol adds additional security features that enable interoperable authentication, delegation and privileges in a CORBA environment. It supports interoperability with the EJB 2.0 specification and can be used with SSL.
- J2EE security uses the security collaborator to enforce J2EE-based security policies and support J2EE security APIs. APIs are accessed from WebSphere applications in order to access security mechanisms and implement security policies. J2EE security guards access to Web resources such as servlets/JSPs and EJB methods based on roles defined by the application developer. Users and groups are assigned to these roles during application deployment.
- IBM Java Secure Socket Extension (JSEE) is the Secure Sockets Layer (SSL) implementation used by WebSphere Application Server. It is a set of Java packages that enable secure Internet communications. It implements a Java version of SSLand Transport Layer Security (TLS) protocols and includes functionality for data encryption, server authentication, message integrity, and client authentication.

WebSphere Application Server V5 security relies on and enhances all the above mentioned layers. It implements security policy in a unified manner for both Web and EJB resources.

Table 2-8 on page 62 shows the security features supported by the WebSphere Application Server configurations.

Table 2-8 WebSphere Application ServerWebSphere Application Server security support

	Express	**Base**	**ND**	**Enterprise**
Java 2 security	No	Yes	Yes	Yes
J2EE security (role mapping)	Yes	Yes	Yes	Yes
JAAS	Yes	Yes	Yes	Yes
CSIv2	No	Yes	Yes	Yes
Security authentication	LDAP, SWAM	LDAP, SWAM	LDAP	LDAP
User registry	Local OS, LDAP, Custom Registry	Local OS, LDAP, Custom Registry	Local OS, LDAP, Custom Registry	Local OS, LDAP, Custom Registry

Figure 2-10 presents a general view of the logical layered security architecture model of WebSphere Application Server. The flexibility of that architecture model lies in pluggable modules which can be configured according to the requirements and existing IT resources.

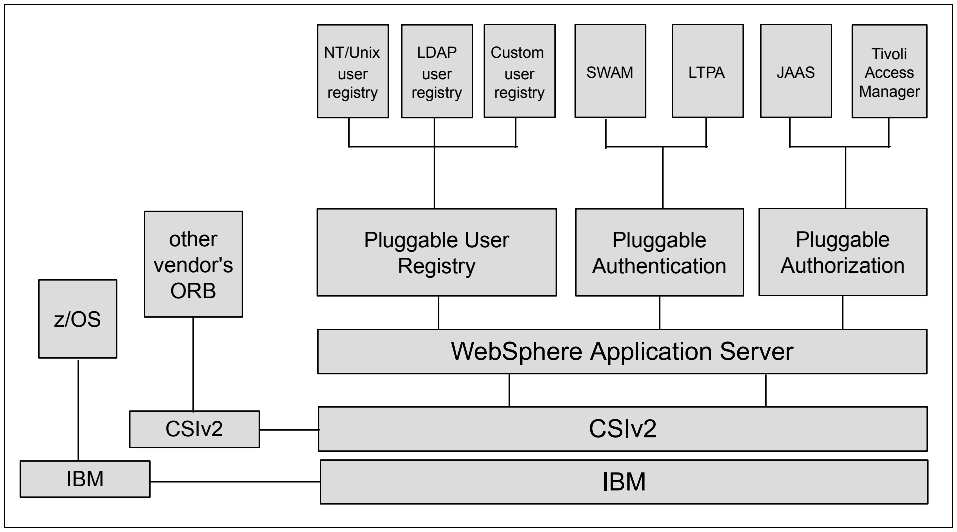


Figure 2-10 WebSphere V5 extensible security architecture

2.10.1 User registry

The pluggable user registry allows you to configure different databases to store user IDs and passwords that are used for authentication and authorization. There are three options available:

- Local operating system user registry

  When configured, WebSphere uses the operating system's users and groups for authentication.

- LDAP user registry

  In many solutions, LDAP user registry is recommended as the best solution for large scale Web implementations. Most of the LDAP servers available on the market are well equipped with security mechanisms that can be used to securely communicate with WebSphere Application Server. The flexibility with which an administrator can set up search parameters to adapt WebSphere to different LDAP schemas is considerable.

- Custom user registry

  This leaves an open door for any custom implementation of a user registry database. WebSphere API provides the UserRegistry Java interface that you should use to write the custom registry. This interface may be used to access virtually any relational database, flat files and so on.

Only one single registry can be active at a time.

2.10.2 Authentication

Authentication is the process of establishing whether a client is valid in a particular context. A client can be either an end user, a machine, or an application. The pluggable authentication module allows you to choose whether WebSphere will authenticate the user or will accept the credentials from external authentication mechanisms.

An authentication mechanism in WebSphere typically collaborates closely with a user registry when performing authentication. The authentication mechanism is responsible for creating a credential which is a WebSphere internal representation of a successfully authenticated client user. Not all credentials are created equal. The abilities of the credential are determined by the configured authentication mechanism.

Although WebSphere provides several authentication mechanisms, only a single "active" authentication mechanism can be configured at once. The active authentication mechanism is selected when configuring WebSphere global security.

WebSphere provides two authentication mechanisms: Simple WebSphere Authentication Mechanism (SWAM) and Lightweight Third Party Authentication (LTPA). These two authentication mechanisms differ primarily in the distributed security features each supports.

- SWAM (Simple WebSphere Authentication Mechanism)

  The SWAM authentication mechanism is intended for simple, non-distributed, single application server type runtime environments. The single application server restriction is due to the fact that SWAM does not support forwardable credentials. What this means is that if a servlet or EJB in application server process 1 invokes a remote method on an EJB living in another application server process 2, the identity of the caller identity in process 1 is not transmitted to server process 2. What is transmitted is an unauthenticated credential, which, depending on the security permissions configured on the EJB methods, may cause authorization failures.

  Since SWAM is intended for a single application server process, single sign-on (SSO) is not supported.

  The SWAM authentication mechanism is suitable for simple environments, software development environments, or other environments that do not require a distributed security solution.

  SWAM relies on the session ID; it is not as secure as LTPA, therefore using SSL with SWAM is strongly recommended.

- LTPA (Light Weight Third Party Authentication)

  Lightweight Third Party Authentication (LTPA) is intended for distributed, multiple application servers and machine environments. It supports forwardable credentials and SSO. LTPA is able to support security in a distributed environment through the use of cryptography. This allows LTPA to encrypt, digitally sign and securely transmit authentication related data and later decrypt and verify the signature.

  LTPA requires that the configured user registry be a central shared repository such as LDAP or a Windows domain type registry.

2.10.3 Authorization

Pluggable authorization interfaces will allow the use of different authorization mechanisms for WebSphere applications. In the current version, JAAS is supported and Tivoli Access Manager is an external authorization system.

WebSphere Application Server standard authorization mechanisms are based on the J2EE security specification and Java Authentication and Authorization Services (JAAS).

- The Java 2 security architecture uses a security policy to specify who is allowed to execute code in the application. Code characteristics, like a code signature, signer ID, or source server, determine whether or not the code will be granted access to be executed.
- JAAS extends this approach with role-based access control. Permission to execute a code is granted not only based on the code characteristics but also on the user, who is running it. JAAS programming models allow the developer to design application authentication in a pluggable fashion, which makes the application independent from the underlying authentication technology.

For each authenticated user, a Subject class is created and a set of Principals is included in the subject in order to identify that user. Security policies are granted based on possessed principals.

2.10.4 Security components

Figure 2-11 shows an overview of the security components that come into play in WebSphere Application Security.

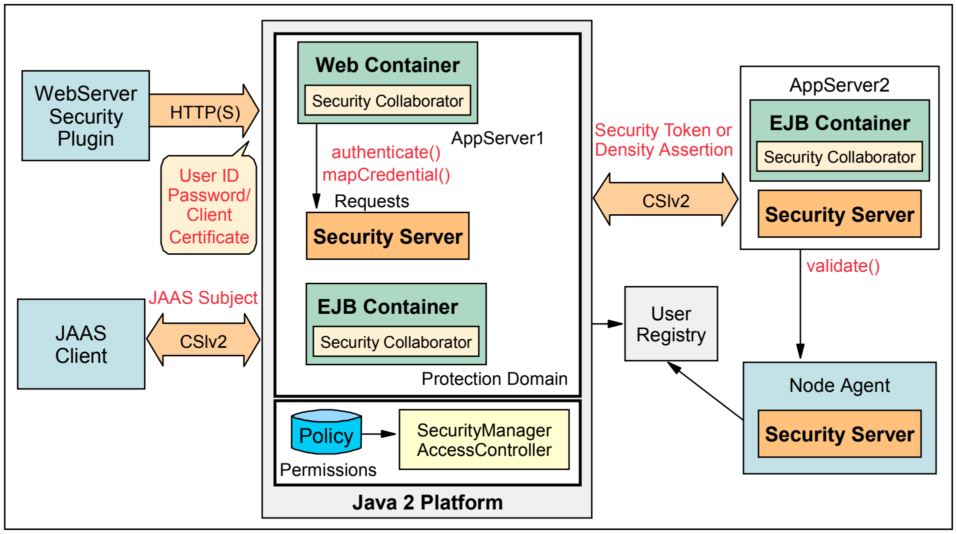


Figure 2-11 WebSphere Application Security components

Security server

The security server is a component of WebSphere Application Server that runs in each application server process. If multiple application server instances are executed on a single node, then multiple security servers exist on that node.

The security server component is responsible for managing authentication and for collaborating with the authorization engine and the user registry.

Security collaborators

Security collaborators are application server processes responsible for enforcing security constraints specified in deployment descriptors. They communicate with the security server every time that authentication and authorization actions are required. The following security collaborators are identified.

- Web security collaborator

  The Web security collaborator resides in the Web container and provides the following services to the application:

  - Checks authentication
  - Performs authorization according to the constraint specified in the deployment descriptor
  - Logs security tracing information

- EJB security collaborator

  The EJB security collaborator resides in the EJB container. It uses CSIv2 and SAS to authenticate Java client requests to enterprise beans. It works with the security server to perform the following functions:

  - Check authorizations according to the specified security constraint
  - Support communication with local user registry
  - Log security tracing information
  - Communicate external ORB using CSIv2 when a request for a remote bean is issued

2.10.5 Security flows

The following sections outline the general security flow.

Web browser communication

The following steps describe the interaction of the components from a security point of view when a Web browser sends a request to a WebSphere application.

1. The Web user requests a Web resource protected by WebSphere Application Server.
2. The Web server receives the request and recognizes that the requested resource is on the application server, and, using the Web server plug-in, redirects the request.

3. The Web server plug-in passes the user credentials to the Web security collaborator, which performs user authentication.
4. After successful authentication, the Web request reaches the Web container. The Web security collaborator passes the user's credentials and the security information contained in the deployment descriptor to the security server for authorization.
5. Upon subsequent requests, authorization checks are performed either by the Web collaborator or the EJB collaborator, depending on what the user is requesting. User credentials are extracted from the established security context.

Administrative tasks

Administrative tasks are issued using either the Web-based administrative console or the wsadmin scripting tool. The following steps illustrate how the administration tasks are executed.

1. The administration client generates a request that reaches the server side ORB and JMX MBeans. The JMX MBeans represent managed resources.
2. The JMX MBeans contact the security server for authentication purposes. JMX beans have dedicated roles assigned and do not use user registry for authentication and authorization.

Java client communication

The steps below describe how a Java client interacts with a WebSphere application.

1. A Java client generates a request that reaches the server side ORB.
2. The CSIv2 or IBM SAS interceptor performs authentication on the server side on behalf of the ORB, and sets the security context.
3. The server side ORB passes the request to the EJB container.
4. After submitting a request to the access protected EJB method, the EJB container passes the request to the EJB collaborator.
5. The EJB collaborator reads the deployment descriptor from the .ear file and user credential from the security context.
6. Credentials and security information is passed to the security server which validates user access rights and passes this information back to the collaborator.
7. After receiving a response from the security server, the EJB collaborator authorizes or denies access to the requested resource for the user.

2.11 Resource providers

Resource providers define resources needed by running J2EE applications. Table 2-9 shows the resource provider support of the WebSphere Application Server configuration.

Table 2-9 WebSphere Application Server resource provider support

	Express	**Base**	**ND**	**Enterprise**
JDBC provider	Yes	Yes	Yes	Yes
JavaMail	Yes	Yes	Yes	Yes
JMS providers	No	Yes	Yes	Yes
Resource environment providers	No	Yes	Yes	Yes
URL providers	No	Yes	Yes	Yes
Resource adapters	No	Yes	Yes	Yes

2.11.1 JDBC resources

A data source represents a real-world data source, such as a relational database. When a data source object has been registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the DataSource object. This makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property in the data source. None of the code using that data source needs to be touched.

Once a data source has been registered with an application server's JNDI name space, application programmers can use it to make a connection to the data source it represents.

The connection will usually be a pooled connection, that is, once the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

Data source classes and JDBC drivers are implemented by the data source vendor. By configuring a JDBC provider, information about the set of classes used to implement the data source and the database driver is provided, that is, it provides the environment settings for the DataSource object.

WebSphere Application Server V5 provides two types of data sources, each differentiated by how the connections are handled:

- ► WebSphere Version 4 data source
- ► WebSphere Version 5 data source

Version 4 data source

WebSphere V4.0 provides its own JDBC connection manager to handle connection pooling and JDBC access. This support is included with WebSphere Application Server V5 to provide support for J2EE 1.2 applications. If an application chooses to use a V4 data source, the application will have the same connection behavior as in WebSphere V4.

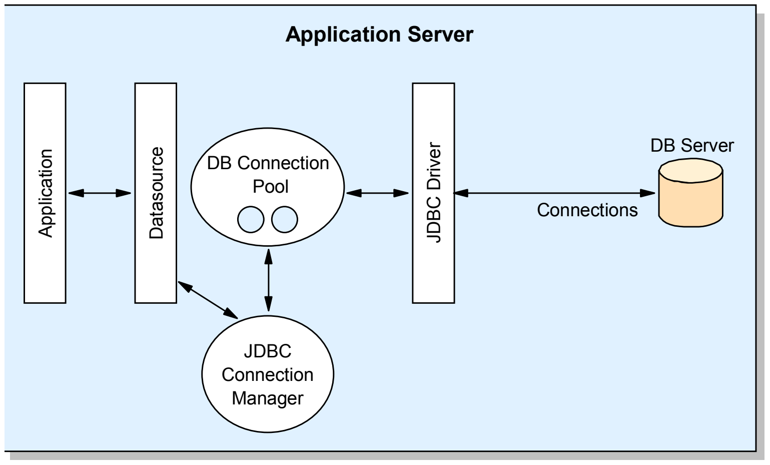


Figure 2-12 Connection pooling in WebSphere V4

WebSphere 5 data source

In WebSphere Application Server V5, connection pooling is provided by two parts, a JCA Connection Manager and a relational resource adapter.

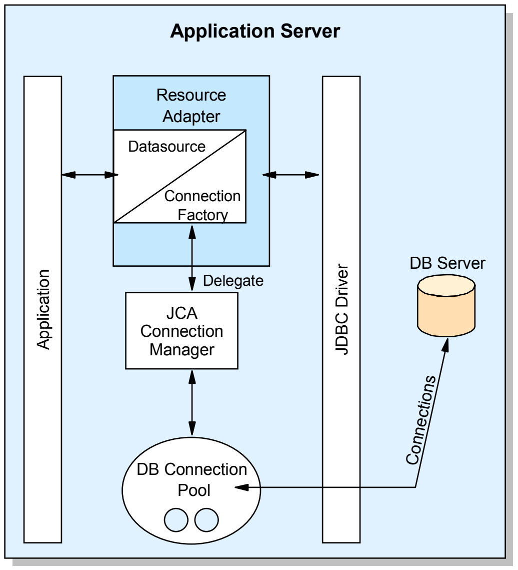


Figure 2-13 Resource adapter in J2EE connector architecture

The JCA Connection Manager provides the connection pooling, local transaction, and security supports. The relational resource adapter provides both JDBC wrappers and JCA CCI implementation that allows BMP, JDBC applications and CMP beans to access the database.

2.11.2 JavaMail

The JavaMail APIs provide a platform and protocol-independent framework for building Java-based mail client applications. The JavaMail APIs require service providers, known in WebSphere as protocol providers, to interact with mail servers that run the pertaining protocols.

A JavaMail provider encapsulates a collection of protocol providers. WebSphere Application Server has a Built-in Mail Provider that encompasses three protocol providers: SMTP, IMAP and POP3. These protocol providers are installed as the default and should be sufficient for most applications.

- Simple Mail Transfer Protocol (SMTP): this is a popular transport protocol for sending mail. JavaMail applications can connect to an SMTP server and send mail through it by using this SMTP protocol provider.
- Post Office Protocol (POP3): this is the standard protocol for receiving mail.
- Internet Message Access Protocol (IMAP): this is an alternative protocol to POP3 for receiving mail.

To use other protocols, you must install the appropriate service provider for those protocols.

In addition to service providers, JavaMail requires the JavaBeans Activation Framework (JAF) as the underlying framework to deal with complex data types that are not plain text, like Multipurpose Internet Mail Extensions (MIME), Uniform Resource Locator (URL) pages, and file attachments.

The JavaMail APIs, the JAF, the service providers and the protocols are shipped as part of WebSphere Application Server using the following Sun licensed packages:

- mail.jar: contains the JavaMail APIs, and the SMTP, IMAP, and POP3 service providers.
- activation.jar: contains the JavaBeans Activation Framework.
- Application Server - Express: supports JavaMail V1.2 and the JAF V1.0.

2.11.3 JCA resource adapters

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EIS), for example ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language.

The JCA resource adapter is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the connectivity between J2EE components (an application server or an application client) and an EIS.

To use a resource adapter, you need to install the resource adapter code and create connection factories that use the adapter.

One resource adapter, the WebSphere Relational Resource Adapter, is predefined for handling data access to relational databases. This resource adapter provides data access through JDBC calls to access databases dynamically. It provides connection pooling, local transaction, and security support. The WebSphere persistence manager uses this adapter to access data for container-managed persistence (CMP) beans.

2.11.4 URL providers

URL providers implement the functionality for a particular URL protocol, such as HTTP, by extending the java.net.URLStreamHandler and java.net.URLConnection classes. It enables communication between the application and a URL resource that is served by that particular protocol.

A URL provider named Default URL Provider is included in the initial WebSphere configuration. This provider utilizes the URL support provided by the IBM JDK. Any URL resource with protocols based on the Java 2 Standard Edition 1.3.1, such as HTTP, FTP or File, can use the default URL provider.

Customers can also “plug in” their own URL providers that implement other protocols not supported by the JDK.

2.11.5 JMS providers

The JMS functionality provided by WebSphere Application Server includes support for three types of JMS provider:

- WebSphere JMS provider (embedded messaging)
- WebSphere MQ JMS provider
- Generic JMS providers

There can be more than one JMS provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of the WebSphere JMS provider, WebSphere MQ JMS provider and a generic JMS provider.

There can only be one JMS server process (embedded WebSphere JMS provider) per node. However, it is possible to have both the WebSphere JMS provider and full WebSphere MQ concurrently installed on the same machine. In this case, there will only be one copy of the JMS client classes (MA88) and MQ transport code (binaries). Because of this, both must be at the same version and patch level.

The support provided by WebSphere administration tools for configuration of JMS providers differs depending upon the provider. Table 2-10 provides a summary of the support.

Table 2-10 WebSphere administration support for JMS provider configuration

Configurable objects	WebSphere JMS provider	WebSphere MQ JMS provider	Generic JMS provider
Initial context factory and provider URL	N **	N **	Y
Messaging system objects (queues/topics)	Y	N	N
JMS administered objects (JMS connection factory and JMS destination)	Y	Y	N
** The settings are not exposed in the WebSphere administrative console.			

WebSphere JMS provider (embedded messaging)

The embedded messaging support is provided as a WebSphere Application Server installation option. The embedded support provides both point-to-point and publish/subscribe functions.

This support, referred to as the WebSphere JMS provider, is composed of the following IBM products:

- WebSphere MQ
- WebSphere MQ MA88 SupportPac™ - JMS client classes
- An early release of WebSphere Event Broker, providing the publish/subscribe service

Each of these is reduced in footprint and function compared to the independently available product.

The WebSphere JMS provider supports the requirements of the J2EE 1.3 specification:

- Support for the JMS 1.0.2 specification.
- Providing an implementation of JMS Server 1.0.
- Full compliance with the J2EE 1.3 compliance tests

The WebSphere JMS provider does not support:

- APIs other than those defined in the JMS API
- Interoperability with WebSphere MQ queue managers

Security is integrated with the WebSphere security system, with user and group access being configured using the WebSphere administration tools.

The JMS server hosts the broker and manages the external WebSphere MQ processes, including creation, management and deletion of the WebSphere MQ queues and topics.

All JMS client access is performed using the WebSphere MQ client (TCP/IP) mode.

WebSphere MQ JMS provider

WebSphere Application Server V5 supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

Generic JMS providers

WebSphere Application Server V5 supports the use of generic JMS providers, as long as they implement the ASF component of the JMS 1.0.2 specification. JMS resources for generic JMS providers are not configurable using WebSphere administration.

2.11.6 Resource environment providers

The java:comp/env environment provides a single mechanism by which both JNDI name space objects and local application environment objects can be looked up. WebSphere Application Server provides a number of local environment entries by default.

The J2EE 1.3 specification also provides a mechanism for defining custom (non-default) environment entries using <resource-env-ref> entries defined in an application's standard deployment descriptors. The J2EE 1.3 specification separates the definition of the resource environment entry from the application by:

1. Requiring the application server to provide a mechanism for defining separate administrative objects that encapsulate a resource environment entry. The administrative objects are to be accessible via JNDI in the application server's local name space (java:comp/env).
2. Specifying the administrative object's JNDI lookup name and the expected returned object type in <resource-env-ref>.

WebSphere Application Server supports the <resource-env-ref> mechanism by providing administration objects for the following:

- Resource environment provider

  Defines an administrative object that groups together the referenceable, resource environment entry administrative objects and any required custom properties.
- Referenceable

  Defines the class name of the factory class that returns object instances implementing a Java interface.
- Resource environment entry

  Defines the binding target (JNDI name), factory class and return object type (via the link to the referenceable) of the resource environment entry.

2.12 Administration

WebSphere Application Server V5 provides a new administration model based on the JMX framework. JMX allows you to wrap hardware and software resources in Java and expose them in a distributed environment. JMX also provides a mapping framework for integrating existing management protocols, such as SNMP, into JMX's own management structures.

Each application server has an administration service that provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository. The repository is actually a set of XML files stored in the server's file system.

2.12.1 Administration tools

Table 2-11 shows the administration tool support for WebSphere Application Server by configuration.

Table 2-11 WebSphere Application Server administration tool support

	Express	Base	ND	Enterprise
Administrative console	Yes	Yes	Yes	Yes
Commands	Yes	Yes	Yes	Yes
wsadmin	Included, but not recommended	Yes	Yes	Yes

Administrative console

The administrative console is a Web browser based interface that provides configuration and operation capability. It is implemented with an enterprise application (adminconsole.ear) installed on an application server. The administrator connects to the application using a Web browser client.

Users assigned to different administration roles can manage the application server and certain components and services using this interface. Because the administrative console is an application, the server and application must both be started before you can use it.

In the Express and Base configurations, the adminconsole application is installed on server1. If you create additional servers, you can configure them using the administrative console application on server1 but will have no runtime capabilities. You can install the adminconsole.ear application on each server in order to have full operational capabilities for that server.

In the Network Deployment and Enterprise configurations, the administrative console application is installed and runs on the Deployment Manager. When a node is added to a cell, the adminconsole application is deleted from the node and the configuration files are integrated into the master cell repository to be maintained by the Deployment Manager.

Commands

WebSphere Application Server provides a set of commands in the <server_install>/bin directory that allow you to perform a subset of administrative functions.

For example, the **`startServer`** command is provided to start an application server.

wsadmin scripting client

The wsadmin scripting client provides extra flexibility over the Web-based administration application, allowing administration using the command-line interface. Using the scripting client not only makes administration quicker, but helps automate the administration of multiple application servers and nodes using scripts.

The scripting client uses the Bean Scripting Framework (BSF), which allows a variety of scripting languages to be used for configuration and control.

The wsadmin scripting interface is included in all WebSphere Application Server configurations but is targeted toward advanced users. The use of wsadmin requires in-depth familiarity with application server architecture and a scripting language. It is not recommended for WebSphere Application Server - Express users.

2.12.2 Configuration repository

The configuration repository holds copies of the individual component configuration documents. Unlike previous versions of WebSphere Application Server, which used a relational database to hold configuration information, in V5 all configuration information is stored in XML files. The application server's Admin service takes care of the configuration and makes sure it is consistent during the runtime.

2.12.3 Centralized administration

The Network Deployment and Enterprise configurations allow multiple servers and nodes to be administered from a central location. This is facilitated through the use of a central Deployment Manager that handles the administration process and distributes the updated configuration to the node agent for each node. The node agent, in turn, is responsible for maintaining the configuration for the servers in the node.

Table 2-12 WebSphere Application Server distributed administration support

	Express	Base	ND	Enterprise
Deployment Manager	No	No	Yes	Yes
Node agent	No	No	Yes	Yes

Managed processes

All operating system processes that are components of the WebSphere product are called managed servers or managed processes. JMX support is embedded in all managed processes and these processes are available to receive administration commands and to output administration information about the state of the managed resources within the processes.

WebSphere provides the following managed servers/processes:

- Deployment Manager

  Provides a single point to access all configuration information and control for a cell. The Deployment Manager aggregates and communicates with all of the node agent processes on each node in the system.

- Node agent

  Aggregates and controls all of the WebSphere managed processes on its node. There is one node agent per node.

- Application server

  Managed server that hosts J2EE applications.

- JMS server

  Managed server that hosts the embedded messaging service for a node. There is one JMS server per node. In a base configuration, the JMS server functions are provided within the application server process.

Deployment Manager

The Deployment Manager process provides a single, central point of administrative control for all elements in the cell. It hosts the Web-based administrative console application. Administrative tools that need to access any managed resource in a cell usually connect to the Deployment Manager as the central point of control.

The Deployment Manager is responsible for the content of the repositories (configuration and application binaries) on each of the nodes. It manages this through communication with the node agent process resident on each node of the cell.

Using the Deployment Manager, horizontal scaling, vertical scaling and distributed applications are all easy to administer and manage, since application servers are managed by nodes, and one or more nodes is/are managed by a cell.

Node agent

The node agent is an administrative process and is not involved in application serving functions. It hosts important administrative functions such as:

- File transfer services
- Configuration synchronization
- Performance monitoring

The node agent aggregates and controls all the managed processes on its node by communicating with:

- The Deployment Manager, to coordinate configuration synchronization and to perform management operations on behalf of the Deployment Manager.
- Application servers and JMS servers, to manage (start/stop) each server and to update its configuration and application binaries as required.

Only one node agent is defined (and run) on each node. In an Express or Base installation, there is no node agent.

Master configuration repository

With the Network Deployment and Enterprise configurations, a master configuration repository that contains all of the cell's configuration data is maintained by the Deployment Manager. The configuration repository at each node is a synchronized subset of the master repository. The node repositories are read-only for application server access, since only the Deployment Manager can initiate their update, pushing configuration changes out from the cell master configuration repository.

2.13 The flow of an application

Figure 2-14 on page 79 shows the typical application flow for Web browser clients using either JDBC (from a servlet) or EJB to access application databases.

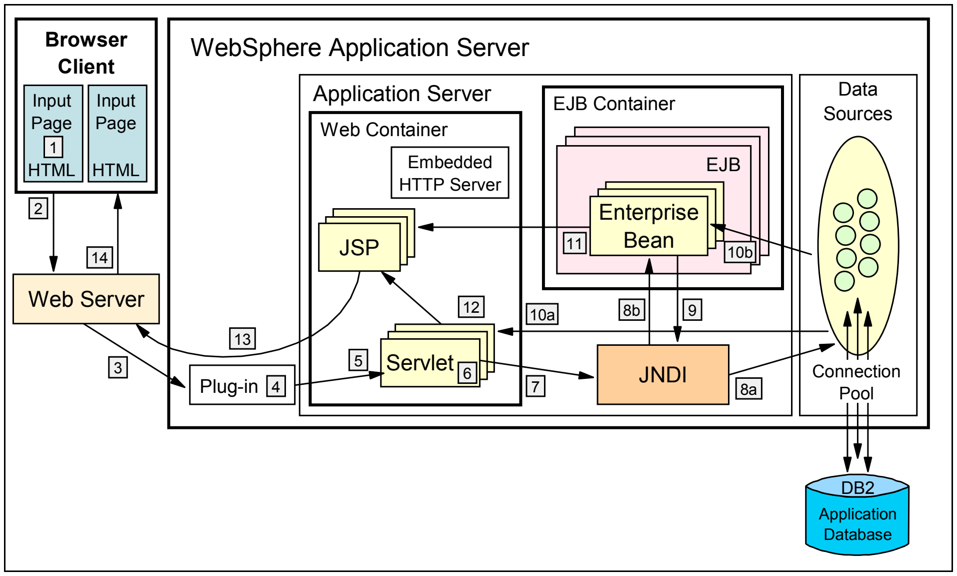


Figure 2-14 Application flow

1. A Web client requests a URL in the browser (input page).
2. The request is routed to the Web server over the Internet.
3. The Web server immediately passes the request to the Web server plug-in. All requests go to the WebSphere plug-in first.
4. The Web server plug-in examines the URL, verifies the list of hostname aliases from which it will accept traffic based on the virtual host information, and chooses a server to handle the request.
5. A stream is created. A stream is a connection to the Web container. It is possible to maintain a connection (stream) over a number of requests. The Web container receives the request and, based on the URL, dispatches it to the proper servlet.
6. If the servlet class is not loaded, the dynamic class loader loads the servlet (servlet *init()*, then *doGet()* or *doPost()*).
7. JNDI is now used for lookup of either datasources or EJBs required by the servlet.
8. Depending upon whether a datasource is specified or an EJB is requested, the JNDI will direct the servlet:
   a. To the corresponding database, and get a connection from its connection pool in the case of a data source

   b. To the corresponding EJB container, which then instantiates the EJB when an EJB is requested

9. If the EJB requested involves an SQL transaction, it will go back to the JNDI to look up the datasource.
10. The SQL statement will be executed and the data retrieved will be sent back:
    a. To the servlet
    b. To the EJB
11. Data beans are created and handed off to JSPs in the case of EJBs.
12. The servlet sends data to JSPs.
13. The JSP generates the HTML that is sent back through the WebSphere plug-in to the Web server.
14. The Web server sends the output page (output HTML) to the browser.

2.14 Developing and deploying applications

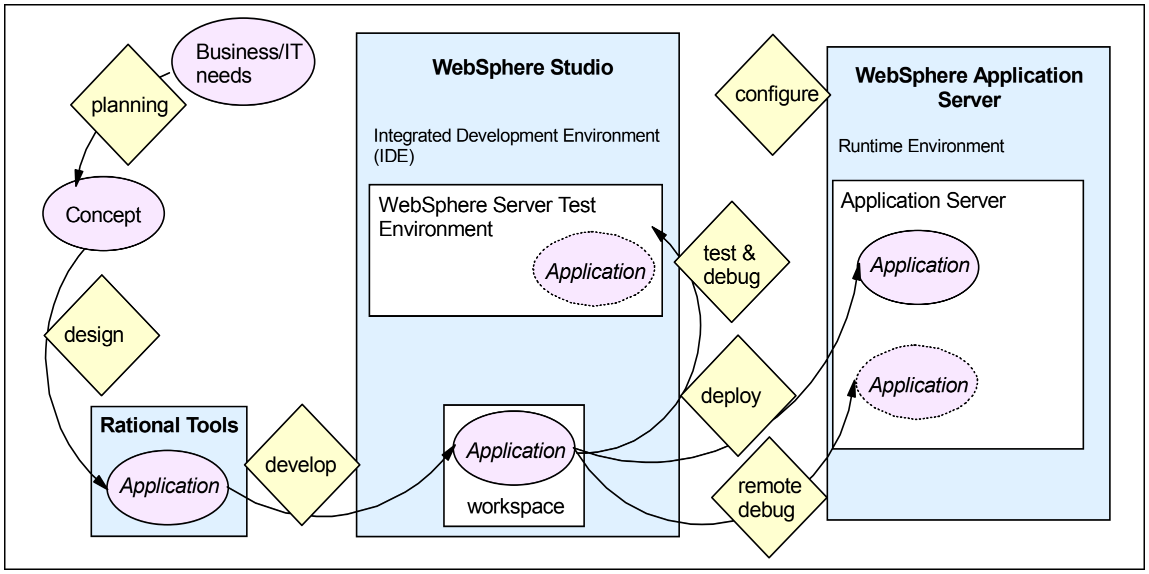


Figure 2-15 Develop and deploy

Application design

Design tools like Rational Rose or Rational XDE™ can be used to model the application using the Unified Modeling Language (UML). The output of the modeling will generally consist of use case scenarios, class diagrams, and starter code generated based on the model.

Application development

Application development is done using WebSphere Studio (or a comparable IDE) to create the enterprise application. Use the WebSphere Studio configuration that most closely matches your needs and the capabilities of the WebSphere Application Server configuration to build the enterprise application.

You can start by importing pre-generated code such as from Rational Rose, a sample application, or an existing production application, or you can start from scratch.

WebSphere Studio provides many tools and aids to get you started quickly. WebSphere Studio supports team development using CVS or Rational ClearCase, allowing multiple developers to share a single master source copy of the code.

During the development phase, component testing can be done using the built-in WebSphere Application Server test environment. WebSphere Studio provides server tools capable of creating and managing servers both in the test environment and on remote server installations. The application is automatically packaged into an EAR file for deployment when you run the application on a server using WebSphere Studio.

Application packaging

J2EE 1.3 applications are packaged into Enterprise Application Archive (EAR) files to be deployed to one or more application servers. A J2EE 1.3 application contains any or all of the modules shown in Table 2-13.

Table 2-13 J2EE 1.3 application modules

Module	Filename	Contents
Web module	<module>.war	Servlets, JSP files, and related code artifacts.
EJB module	<module>.jar	Enterprise beans and related code artifacts.
Application client module	<module>.jar	Application client code.
Resource adapter module	<module>.rar	Library implementation code that your application uses to connect to enterprise information systems (EIS).

This packaging is done automatically in WebSphere Studio when you export an application for deployment. If you are using another IDE, WebSphere Application

Server (with the exception of the Express configuration) provides the Assembly Toolkit in the Application Server Toolkit for packaging applications.

Application deployment

Applications are installed on application servers using the administrative console or the wsadmin scripting interface. An application can be deployed to a single server or a cluster. In the case of a cluster, it is installed on each application server in the cluster.

Installing an application involves the following:

- Binding resource references (created during packaging) to actual resources. For example, a data source would need to be bound to a real database.
- Defining JNDI names for EJB home objects.
- Specifying data source entries for entity beans.
- Binding EJB references to the actual EJB JNDI names.
- Mapping Web modules to virtual hosts.
- Specifying listener ports for message-driven beans.
- Mapping application modules to application servers.
- Mapping security roles to users or groups.

After a new application is deployed, the Web server plug-in configuration file needs to be regenerated and copied to the Web server.

Part 2

Installing WebSphere

This part takes you through the process of planning and installing a WebSphere environment for the Windows and AIX platforms.

- For information about installing WebSphere on the Linux operating system, see *IBM WebSphere V5.0 for Linux, Implementation and Deployment Guide*, REDP3601 at `http://www.ibm.com/redbooks`.
- For information about installing WebSphere on the OS/400 operating system, see *WebSphere Installation, Configuration, and Administration in an iSeries Environment*, SG24-6588.

3

Topology selection

This chapter describes various IBM WebSphere Application Server Network Deployment topologies. In addition, options regarding the use of application server clusters, multiple cells, vertical and horizontal scaling, as well as multiple tiers (separating the Web containers and EJB containers) are covered.

This chapter should be used as a basis for determining what the overall layout of your WebSphere environment should be. The installation procedures covered later in this book assume that you have chosen a topology.

3.1 Topology selection criteria

A variety of factors come into play when considering the appropriate topology, or configuration, for a WebSphere deployment. The selection criteria typically include a review of your requirements in the following areas:

- Security
- Performance
- Throughput
- Scalability
- Availability
- Maintainability
- Session management

This section reviews these factors, their availability, and support provided by IBM WebSphere Application Server Network Deployment.

3.1.1 Security

Security concerns usually require a physical separation of the Web server from the application server processes, typically across one or more firewalls. A common configuration would be the use of two firewalls to create a demilitarized zone (DMZ). Information in the DMZ has some protection based on protocol filtering between the Internet and the DMZ. A Web server intercepts the requests and forwards them on through the next firewall. The sensitive portions of the application and business data reside behind the second firewall, which filters based on IP addresses or domains.

3.1.2 Performance

Performance involves minimizing the response time for a given transaction load. Although a number of factors relating to application design can affect performance, one or both of the following techniques are commonly used:

- Vertical scaling

  Involves creating additional application server processes on a single physical machine, providing for software/application server failover as well as load balancing across multiple JVM (application server processes). Vertical scaling allows an administrator to profile an existing application server for bottlenecks in performance, and potentially use additional application servers, on the same machine, to get around these performance issues.

- Horizontal scaling

  Involves creating additional application server processes on multiple physical machines to take advantage of the additional processing power available on each machine. This provides hardware failover support and allows an administrator to spread the cost of an implementation across multiple physical machines.

3.1.3 Throughput

Throughput, while related to performance, involves the creation of a number of application server instances to service the same requests. IBM WebSphere Application Server Network Deployment provides clusters, which logically group a number of application servers. The application servers are added through vertical and/or horizontal scaling

.

Note: There is no longer the concept of server groups and clones, as in WebSphere Application Server V4. Now each application server cooperates in a *cluster*, as a *cluster member*.

3.1.4 Scalability

Multiple machines can be configured to add processing power, improve security, maximize availability, and balance workloads. The components that provide functions for configuring scalability include:

- WebSphere Application Server cluster support

  Clusters allow the creation of a logical group of servers that all host and run the same application(s). Members of a cluster can be located on the same machine (vertical scaling) and/or across multiple machines (horizontal scaling).

  The use of application server clusters can improve the performance of a server, simplify its administration, and enable the use of workload management; however, an administrator will need to assess whether the cost of implementing a clustered configuration (network traffic, performance) outweighs the benefits of a clustered environment. For more details, see *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.

- WebSphere workload management (WLM)

  Incoming processing requests from clients are transparently distributed among the clustered application servers. WLM enables both load balancing and failover, improving the reliability and scalability of WebSphere applications.

Note: IBM WebSphere Application Server Network Deployment provides three types of workload management (WLM):

- Web server WLM

  Uses the Load Balancer feature from Edge Components as an IP sprayer to distribute HTTP requests across Web servers.

- Web server plug-in WLM

  Distributes servlet requests across Web containers.

- Enterprise Java Services (EJS) WLM

  Distributes EJB requests across EJB containers.

- IP sprayer

  Transparently redirects incoming HTTP requests from Web clients to a group of Web servers. Although the clients behave as if they are communicating directly with a given Web server, the IP sprayer is actually intercepting all requests and distributing them among all the available Web servers in the group. IP sprayers (such as the Load Balancer component of Edge Components or Cisco Local Director) can provide scalability, load balancing, and failover for Web servers.

3.1.5 Availability

To avoid a single point of failure and maximize system availability, the topology must have some degree of process redundancy. High-availability topologies typically involve horizontal scaling across multiple machines. Vertical scaling can improve availability by creating multiple processes, but the machine itself becomes a point of failure.

In addition, an IP sprayer can direct client HTTP requests to available Web servers, bypassing any that are offline. Another server can back up the IP sprayer, which eliminates it as a single point of failure.

Improved availability is one of the key benefits of scaling up to multiple machines. IBM WebSphere Application Server Network Deployment provides tools that let you manage availability by distributing critical functionality across multiple machines. Applications hosted on multiple machines generally have less down time and are able to service client requests more consistently.

The following commonly used techniques can be combined to take advantage of the best features of each and create a highly available system.

Hardware and process redundancy

Eliminate the single points of failure in a system by including hardware and process redundancy:

- Use horizontal scaling to distribute application servers across multiple physical machines. If a hardware or process failure occurs, cluster application servers are still available to handle client requests. Web servers and IP sprayers can also benefit from horizontal scaling.
- Use backup servers for databases, Web servers, IP sprayers, and other important resources. This ensures that they remain available if a hardware or process failure occurs.
- Deploy an application in multiple cells. If an entire cell goes offline, the other cells are still available to handle client requests.

Process isolation

Provide process isolation so that failing servers do not negatively impact the remaining healthy servers of a configuration. The following configurations provide some degree of process isolation:

- Deploy the Web server onto a different machine from the application servers. This ensures that problems with the application servers do not affect the Web server, and vice versa.
- Use horizontal scaling, which physically separates application server processes onto different machines.
- Deploy an application in multiple cells. Problems are confined to one cell while the other cells remain available.

Load balancing

Use load-balancing techniques to make sure that individual servers are not overwhelmed with client requests. These techniques include:

- Use of an IP sprayer to distribute requests to the Web servers in the configuration.
- Directing requests from high-traffic URLs to more powerful servers.

The Edge Components included with IBM WebSphere Application Server Network Deployment provide these features.

Failover support

The application must continue to process client requests when servers are stopped or restarted. Ways to provide failover support include:

- Use horizontal scaling with workload management to take advantage of its failover support.

- Use the HTTP transport to distribute client requests among application servers.

Hardware-based high availability

In most cases there is very little to be gained by configuring WebSphere to work in conjunction with a hardware-based high-availability product, for example HACMP, Sun Cluster or MC/ServiceGuard.

The only case where a hardware-based HA solution would provide value is where WebSphere serves as the coordinator of a distributed (two-phase commit) transaction. In all other cases, the cluster capabilities inherent to Network Deployment should be sufficient to provide high availability of the WebSphere runtime.

3.1.6 Maintainability

The topology affects the ease with which system hardware and software can be updated. For instance, using multiple WebSphere cells or horizontal scaling can make a system easier to maintain because individual machines can be taken offline without interrupting other machines running the application.

Sometimes maintainability conflicts with other topology considerations. For example, limiting the number of application server instances makes the application easier to maintain but can have a negative effect on throughput, availability, and performance.

When deciding how much vertical and/or horizontal scaling is to be used in a topology, consideration needs to be made for hardware upgrades, for example, adding or upgrading CPUs and memory.

3.1.7 Session management

Unless only a single application server is used, or the application is completely stateless, maintaining session state between HTTP client requests will also be a factor in determining the chosen topology.

Network Deployment provides two different functions for the sharing of sessions between multiple application server processes:

- Database persistence

  Session data is persisted to a database shared by the application servers.

- Memory-to-memory replication

  Provides replication of session data between the memories of different application server JVMs. A JMS publish/subscribe mechanism, called WebSphere Internal Messaging, is used to provide assured session replication between the JVM processes, by leveraging the internal JMS provider provided with WebSphere Application Server.

  This means that a database product is not required for persistence.

  Memory-to-memory replication is faster, by virtue of the high-performance messaging implementation used by WebSphere V5. However, it is only available in the Network Deployment environment.

  Note: This mechanism is new to IBM WebSphere Application Server Network Deployment as of V5.0. Prior versions of WebSphere provided persistence of the session to a database only.

In addition, the configuration of any IP sprayers (such as provided with the Edge Components) needs to be considered when session state is a factor.

3.1.8 Topology selection criteria summary

The preceding factors are not mutually exclusive. They can be combined in many different ways, as shown by the examples in “Topologies” on page 94.

Table 3-1 provides a summary of the topology selection criteria available with IBM WebSphere Application Server Network Deployment.

Table 3-1 Topology selection summary

Topology	Security	Performance	Throughput	Maintainability	Availability	Session
Vertical scaling		Limited benefit	Limited to resources on a single machine	Easiest to maintain	Process isolation	Required
Horizontal scaling		Best in general	Best in general	Code migration to multiple nodes	Process and hardware redundancy	Required
HTTP separation	Allow for firewalls and DMZs	Usually better than local	Usually better than local			

Topology	Security	Performance	Throughput	Maintainability	Availability	Session
Three tiers	Most options for firewalls	Typically slower than single JVM API	Clustering can improve throughput			
One cell				Ease of maintenance		
Multiple cells				Harder to maintain than single cell	Process, hardware and software redundancy	

3.2 WebSphere component coexistence

IBM WebSphere Application Server Network Deployment supports a number of scenarios for the coexistence of IBM WebSphere Application Server installations on a single machine. This support provides new options to consider when selecting a topology:

- Multiple WebSphere Application Server instances.
- Multiple server instances using a single installation.
- Coexistence of WebSphere Application Server and Network Deployment.
- Single versus multiple Web servers.

3.2.1 Multiple WebSphere instances

IBM WebSphere Application Server supports the following types of product coexistence:

- Multiple instances from a single installation.
- Multiple installations on a single machine. (Note that on Windows, only the last installation is reflected in the Windows registry).

Table 3-2 Multiple installation scenarios

Installed product	with WebSphere Application Server V5 base	with WebSphere Application Server V5 ND
WebSphere Application Server V5 base	Supported Port conflicts can be resolved at installation time.	Supported There are no issues. There are no port conflicts in the default configuration settings for both products. The WebSphere development community commonly uses this environment.
WebSphere Application Server Network Deployment	Supported	Supported Port conflicts can be resolved at installation time.

3.2.2 Multiple server instances using a single installation

The following configurations are supported for running multiple runtime instances using a single installation:

- Creating and running multiple WebSphere Application Server instances from a single WebSphere Application Server installation.
- Creating and running multiple Deployment Manager instances from a single WebSphere Application Server Network Deployment installation.

Although multiple Deployment Managers can be configured for a single Network Deployment installation, these Deployment Managers cannot provide failover or clustering support for each other. In essence this configuration allows the Deployment Managers of multiple, unconnected cells to be configured and run on a single machine.

3.2.3 Coexistence of WebSphere Application Server and Network Deployment

Network Deployment software can be installed on the same machine as WebSphere Application Server. The advantages to this are:

- There is no need for a dedicated machine to host the cell Deployment Manager and its master cell repository.
- Ability to reuse existing backup facilities provided for the node machine.

On the down side, if something happens to either the WebSphere Application Server or Deployment Manager installation that would require the node to be rebuilt, the other component would need to be moved. There is also the possibility that having both would cause contention for system resources.

3.2.4 Single versus multiple Web servers

In addition to multiple instances, WebSphere Application Server also provides Web server options when there are coexisting application servers on a single machine:

- Single Web server for coexisting, multi-version application servers on one machine.
- Single Web server for multiple instances of WebSphere Application Server V5.
- Separate Web servers for each application server instance, when running multiple instances of WebSphere Application Server V5.

3.3 Topologies

IBM WebSphere Application Server Network Deployment supports a number of common topologies:

- Vertical scaling
- HTTP server separation
- Reverse proxy
- Horizontal scaling with clusters
- Horizontal scaling with IP sprayer
- Multiple WebSphere cells
- Multiple clusters on a node
- Combined

3.3.1 Terminology

The illustrations used to represent each topology will include boxes that represent logical runtime functions. The terminology used is taken from the Patterns for e-business Web site at:

http://www.ibm.com/developerworks/patterns/

Note that in the Patterns for e-business, a runtime node represents a logical function. That “node” may actually be implemented using multiple physical machines. Do not confuse that with the concept of a physical node in WebSphere.

Web application server node

A Web application server node is an application server that includes an HTTP server (also known as a Web server) and is typically designed for access by HTTP clients and to host both presentation and business logic.

The Web application server node is a functional extension of the informational (publishing-based) Web server. It provides the technology platform and contains the components to support access to both public and user-specific information by users employing Web browser technology. For the latter, the node provides robust services to allow users to communicate with shared applications and databases. In this way, it acts as an interface to business functions, such as banking, lending, and human resources systems.

The node can contain these data types:

- HTML text pages, images, multimedia content to be downloaded to the client browser
- Servlets, JavaServer Pages
- Enterprise beans
- Application program libraries, such as Java applets for dynamic download to client workstations

In these topologies the Web application server node is implemented using one (or both) of the following:

- WebSphere Application Server and the embedded HTTP transport
- WebSphere Application Server, a supported Web server (for example, IBM HTTP Server), and the Web server plug-in.

Web server redirector node and application server node

Most of the topologies you see will feature a stand-alone Web server residing in a DMZ (between two firewalls). In order to separate the Web server function from the Web application server function, we effectively split the Web application server into two new nodes: a Web server redirector node, and an application server node.

Web clients send requests to the Web server redirector node, which serves the static content such as HTTP pages. Requests for dynamic content, requiring processing by servlets, JSPs, enterprise beans, and back-end applications are forwarded to the application server.

In these topologies, the Web server redirector is implemented using a supported Web server and the appropriate Web server plug-in. The application server node is implemented using WebSphere Application Server.

Domain and protocol firewall nodes

A firewall is a hardware/software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets, and can block some virus attacks - as long as those viruses are coming from the Internet. A firewall can separate two or more parts of a local network to control data exchange between departments. Components of firewalls include filters or screens, each of which controls transmission of certain classes of traffic. Firewalls provide the first line of defense for protecting private information, but comprehensive security systems combine firewalls with encryption and other complementary services, such as content filtering and intrusion detection.

Firewalls control access from a less trusted network to a more trusted network. Traditional implementations of firewall services include:

- Screening routers (the protocol firewall)
- Application gateways (the domain firewall)

A pair of firewall nodes provides increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is typically implemented as an IP router.

Directory and security services node

The directory and security services node supplies information on the location, capabilities, and attributes (including user ID/password pairs and certificates) of resources and users known to this Web application system. This node can supply information for various security services (authentication and authorization) and can also perform the actual security processing, for example, to verify certificates. The authentication in most current designs validates the access to the Web application server part of the Web server, but this node also authenticates for access to the database server.

In these topologies, the directory and security services node is implemented using the IBM Directory Server 4.1.

Web presentation server node

The Web presentation server node provides services to enable a unified user interface. It is responsible for all presentation-related activity. In its simplest form, it serves HTML pages and runs servlets and JSPs. For more advanced patterns, it acts as a portal and provides the access integration services (single sign-on, for example). It interacts with the personalization server node to customize the presentation based on the individual user preferences or on the user role. The Web presentation server allows organizations and their users to standardize and configure the presentation of applications and data in the most efficient way, while enabling fine-grained access control.

In these topologies, the Web presentation server is implemented using WebSphere Application Server. Function is basically limited to that which happens in the Web container. Enterprise beans reside on an application server node.

Database server node

The function of this node is to provide a persistent data storage and retrieval in support of the user-to-business transactional interaction. The data stored is relevant to the specific business interaction, for example bank balance, insurance information, and current purchase by the user.

It is important to note that the mode of database access is perhaps the most important factor determining the performance of this Web application, in all but the simplest cases. The recommended approach is to collapse the database accesses into single or very few calls. One approach for achieving this is by coding and invoking stored procedure calls on the database.

Deployment Manager

In a Network Deployment environment, the Deployment Manager is the focal point for configuration and operation. Though not technically a defined Patterns for e-business runtime node, it is important to this discussion to depict this node in the topologies.

For each of the topologies, a decision needs to be made regarding the placement of the Deployment Manager and master cell repository. The Deployment Manager can be located on a dedicated machine or it can co-exist with a WebSphere Application Server installation. We would recommend placing it on a dedicated machine.

Load balance node

The load balancer node provides horizontal scalability by dispatching HTTP requests among several identically configured Web server or Web server redirector nodes. Using a load balancer may introduce the need to ensure session affinity.

In these topologies, the load balancer node is implemented using the Edge Components.

3.3.2 Topology 1: vertical scaling

Vertical scaling refers to configuring multiple application servers on a single machine, usually by creating a cluster of associated application servers all hosting the same application(s). This configuration is depicted in Figure 3-1.

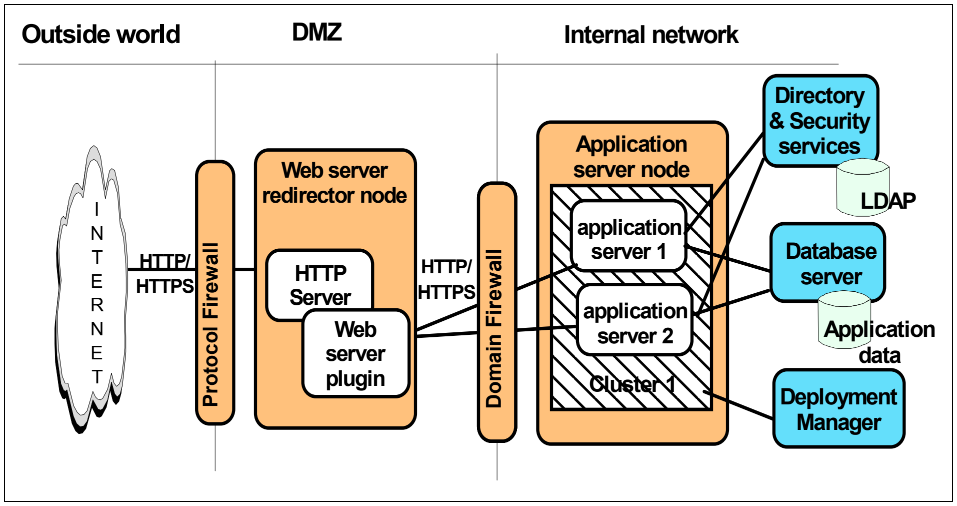


Figure 3-1 Vertical scaling with clusters

The figure represents a simple vertical scaling example, with multiple cluster members on the application server node. Vertical scaling can also be implemented multiple machines in a configuration. Vertical scaling can be combined with other topologies to boost performance and throughput.

Advantages

Vertical scaling has the following advantages:

- Efficient use of machine processing power.

  An instance of an application server runs in a single Java virtual machine (JVM) process. However, the inherent concurrency limitations of a JVM process prevent it from fully utilizing the processing power of a machine. Creating additional JVM processes provides multiple thread pools, each corresponding to the JVM associated with each application server process. This avoids concurrency limitations and lets the application server use the full processing power of the machine.

- Load balancing.

  Vertical scaling topologies can make use of the WebSphere workload management.

- Process failover.

  Vertical scaling can provide failover support among application servers of a cluster. If one application server instance goes offline, the other instances on the machine continue to process client requests.

Disadvantages

Single machine vertical scaling topologies have the drawback of introducing the host machine as a single point of failure in the system.

3.3.3 Topology 2: HTTP server separation

HTTP server separation topologies physically separate the HTTP (Web) server from the application servers, typically to place the HTTP server in a DMZ. Using a DMZ provides an additional layer of security for back-end servers and data.

WebSphere Application Server provides a Web server plug-in for use on the Web server machine. The Web server plug-in routes requests to application servers on remote machines using the HTTP or HTTPS protocol. This configuration is depicted in Figure 3-2.

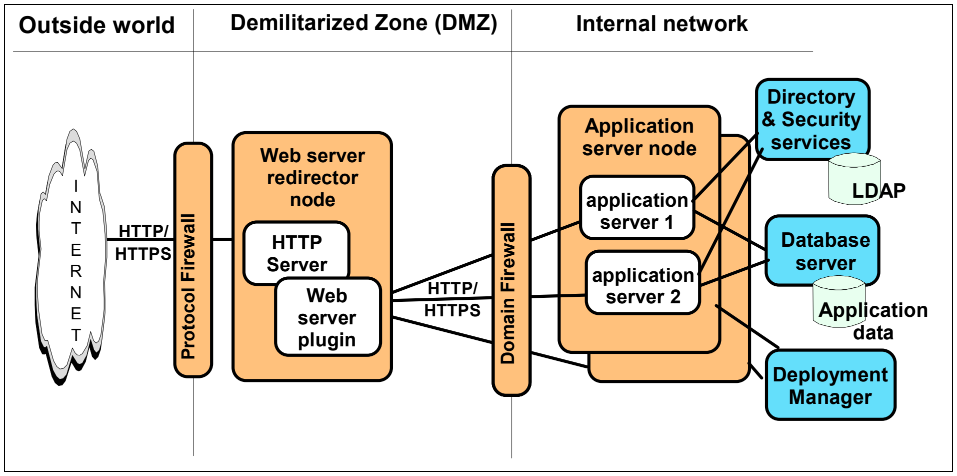


Figure 3-2 HTTP transport remote Web server topology

Variations on this configuration include vertical and horizontal scaling of the application servers.

Advantages

HTTP server separation has the following advantages:

- Supports load balancing and failover, eliminating single points of failure.

  A point of failure exists when one process or machine depends on another process or machine. A single point of failure is undesirable because if the point fails, the whole system will become unavailable. When comparing DMZ solutions, a single point of failure refers to a single point of failure between the Web server and application server. Various failover configurations can minimize downtime and possibly even prevent a failure. However, these configurations usually require additional hardware and administrative resources.

- Avoids data access from DMZ.

  A DMZ configuration protects application logic and data by creating a demilitarized zone between the public Web site and the servers and databases where this valuable information is stored. Desirable DMZ topologies do not have servers that directly access databases from the DMZ.

  Supports WebSphere security in that the location of the Web server is not relevant to the security services provided by WebSphere.

  WebSphere security protects applications and their components by enforcing authorization and authentication policies. Configuration options compatible with WebSphere security are desirable because they do not necessitate alternative security solutions.

- Supports Network Address Translation (NAT) firewalls.

  A firewall product that runs NAT receives packets for one IP address, and translates the headers of the packet to send the packet to a second IP address. In environments with firewalls employing NAT, avoid configurations involving complex protocols in which IP addresses are embedded in the body of the IP packet, such as Java Remote Method Invocation (RMI) or Internet Inter-Orb Protocol (IIOP). These IP addresses are not translated, making the packet useless.

- Supports Secure Sockets Layer (SSL) encryption for communications between the Web server and the application server.

  Configurations that support encryption of communication between the Web server and application server reduce the risk that attackers will be able to obtain secure information by "sniffing" packets sent between the Web server and application server. A performance penalty usually accompanies such encryption.

- Performance bottlenecks may be reduced.

- Administration is simplified. The Web server plug-in uses a single, easy-to-read XML configuration file.

  Some solutions require little or no maintenance after you establish them, while others require periodic administrative steps, such as stopping a server and starting it again after modifying resources that affect the configuration.

Disadvantages

HTTP server separation has the following disadvantages:

- The link between the Web server and WebSphere Application Server is done using a Web server plug-in. The plug-in must be configured after certain configuration changes from the WebSphere Application Server and manually moved to the proper location on the Web server.
- There is no protocol shift for inbound and outbound traffic across a firewall. The Web server sends HTTP requests to application servers behind firewalls and an HTTP port in the firewall must be open to let the requests through.

  Configurations that require switching to another protocol (such as IIOP), and opening firewall ports corresponding to the protocol, are often more complex to set up, and the protocol switching overhead can impact performance. The following approaches can be used to address this:

  – Configure the Web server plug-in to use HTTPS. This will provide a high-security connection between the HTTP server and the application server.

    This connection can be configured so that the Web server plug-in and application server must mutually authenticate each other using public-key infrastructure (PKI).

  – Use different inbound (browser to HTTP server) and outbound (Web server plug-in to application server) port numbers.

Table 3-3 contains a summary of HTTP server separation topology characteristics.

Table 3-3 Characteristics of HTTP server separation topology

Characteristic	**Comment**
SSL support	Yes
Workload management	Yes
Network Address Translation (NAT)	Yes
Performance	High
Administration of configuration	Manual

Characteristic	Comment
Avoids data access from DMZ	Yes
Avoids DMZ protocol switch	No
Avoids single point of failure	Yes
Compatible with WebSphere security	Yes
Minimum required number of firewalls holes	1 per application server, plus 1 if WebSphere Application Server V5 security is used by the Web server.

3.3.4 Topology 3: reverse proxy

Reverse proxy (or IP-forwarding) topologies use a reverse proxy server, such as the one in Edge Components, to receive incoming HTTP requests and forward them to a Web server. The Web server in turn forwards the requests to the application servers that do the actual processing. The reverse proxy returns requests to the client, hiding the originating Web server. This configuration is depicted in Figure 3-3 on page 103.

Note:

- The reverse proxy approach for separating the HTTP server and the application server is generally not suggested for use with WebSphere Application Server V5. The Web server plug-in behaves much like a reverse HTTP proxy, without its disadvantages.
- Use of a reverse proxy has benefits if placed before the HTTP server, as shown in Figure 3-3.

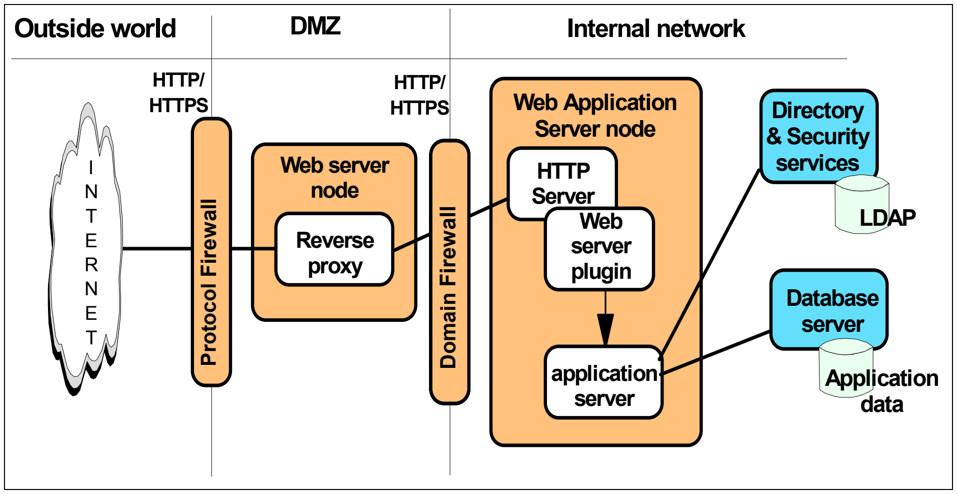


Figure 3-3 Reverse proxy Web server topology

In this example, a reverse proxy resides in a demilitarized zone (DMZ) between the outer and inner firewalls. It listens on an HTTP port (typically port 80) for HTTP requests. The reverse proxy then forwards those requests to an HTTP server that resides on the same machine as application server. After the requests are fulfilled, they are returned through the reverse proxy to the client, hiding the originating Web server.

Reverse proxy servers are typically used in DMZ configurations to provide additional security between the public Internet and the Web servers (and application servers) servicing requests. A reverse proxy product used with WebSphere Application Server must support Network Address Translation (NAT).

Reverse proxy configurations support high-performance DMZ solutions that require as few open ports in the firewall as possible. The reverse proxy capabilities of the Web server inside the DMZ require as few as one open port in the second firewall (potentially two if using SSL - port 443).

Advantages

Advantages of using a reverse proxy server in a DMZ configuration include:

- This is a well-known and tested configuration, resulting in less customer confusion than other DMZ configurations.
- It is a reliable and fast-performing solution.
- It eliminates protocol switching by using the HTTP protocol for all forwarded requests.

- It has no effect on the configuration and maintenance of a WebSphere application.

Disadvantages

Disadvantages of using a reverse proxy server in a DMZ configuration include:

- Requires more hardware and software than similar topologies that do not include a reverse proxy server, making it more complicated to configure and maintain.
- The reverse proxy does not participate in WebSphere workload management.
- It can't be used in environments where security policies prohibit the same port or protocol being used for inbound and outbound traffic across a firewall.

Table 3-4 contains a summary of reverse proxy topology characteristics.

Table 3-4 Characteristics of reverse proxy topology

Characteristic	Comment
SSL support	Yes
Workload management (WLM)	No
Network Address Translation (NAT)	Yes
Performance	High
Administration of configuration	Manual
Avoids data access from DMZ	Yes
Avoids DMZ protocol switch	Yes
Avoids single point of failure	No
Compatible with WebSphere security	Depends on Web server
Minimum required number of firewall holes	1

3.3.5 Topology 4: horizontal scaling with clusters

Horizontal scaling exists when the members of an application server cluster are located across multiple physical machines. This lets a single application span several machines, yet still present a single logical image. This configuration is depicted in Figure 3-4 on page 105.

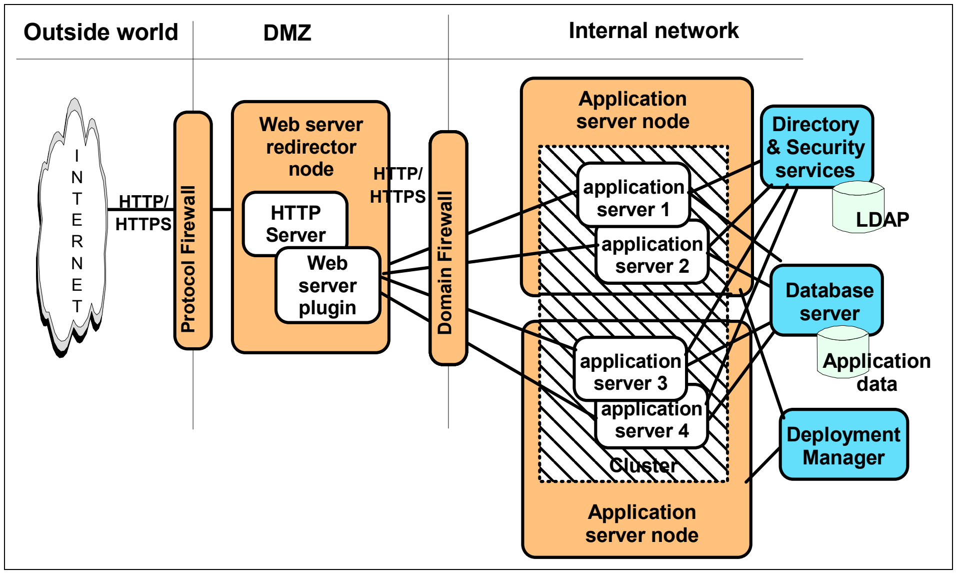


Figure 3-4 Horizontal scaling with clusters topology

The Web server plug-in distributes requests to cluster member application servers on the application server nodes.

The Network Dispatcher component of Edge Components, which can distribute client HTTP requests, can be combined with clustering to reap the benefits of both types of horizontal scaling. See 3.3.6, “Topology 5: horizontal scaling with IP sprayer” on page 106 for a description of this configuration.

Advantages

Horizontal scaling using clusters has the following advantages:

- Provides the increased throughput of vertical scaling topologies but also provides failover support. This topology allows handling of application server process failure and hardware failure without significant interruption to client service.
- Optimizes the distribution of client requests through mechanisms such as workload management or remote HTTP transport.

Disadvantages

Horizontal scaling using clusters has the following disadvantage:

- Increased maintenance effort.

3.3.6 Topology 5: horizontal scaling with IP sprayer

Load-balancing products can be used to distribute HTTP requests among application server instances that are running on multiple physical machines.

The Network Dispatcher component of Edge Components is an IP sprayer that performs intelligent load balancing among Web servers using server availability, capability, workload, and other criteria.

Simple IP sprayer topology

Figure 3-5 depicts a simple horizontal scaling configuration that uses an IP sprayer on the load balancer node to distribute requests among application servers on multiple machines:

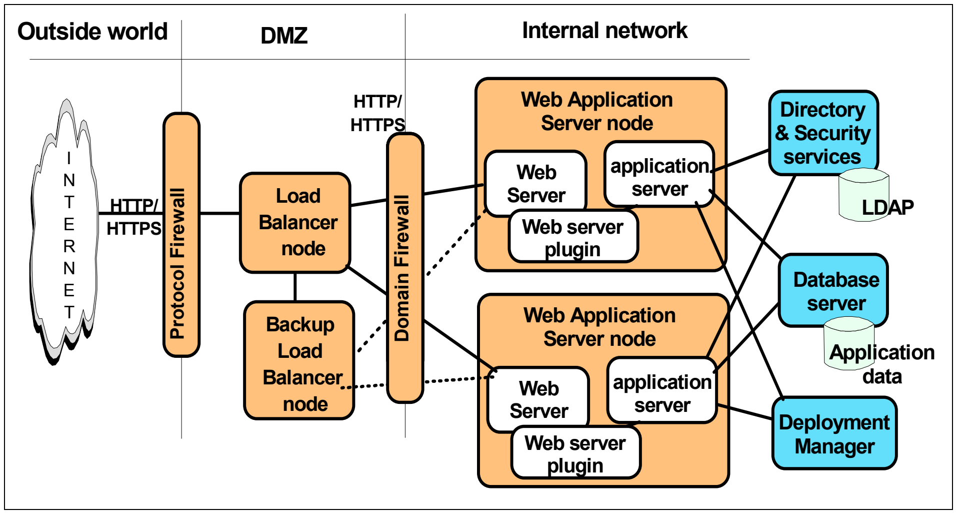


Figure 3-5 Simple IP sprayer-based horizontally scaled topology

A backup node for the load balancer node is normally configured in order to eliminate it as a single point of failure.

Note: In an IP sprayer (or similarly configured) topology, each Web server can be configured to perform workload management over all application servers in a specific cluster. In this configuration, session affinity will be preserved, no matter which Web server is passed the request by the IP sprayer.

Complex IP sprayer topology

Figure 3-6 on page 107 depicts a topology where an IP sprayer distributes requests among several machines containing Web servers and clustered application servers.

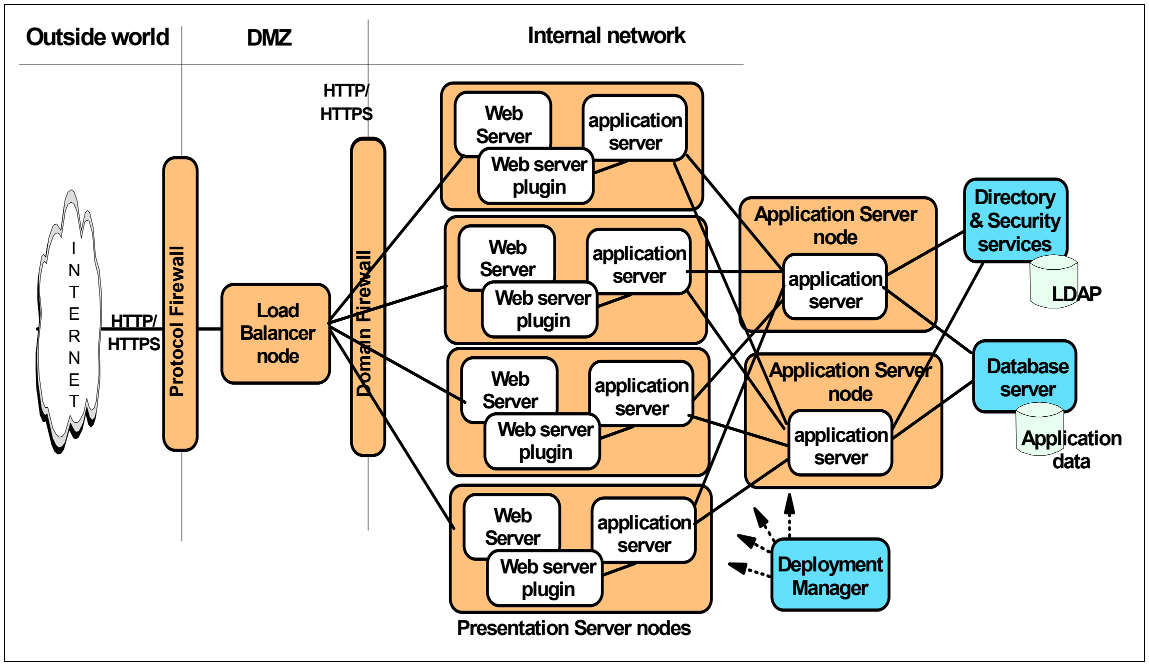


Figure 3-6 Complex IP sprayer horizontally scaled topology

The presentation server nodes host servlet-based applications. The application server nodes contain enterprise beans that access application data and execute business logic. This allows numerous less powerful machines to be employed in the first tier and fewer but more powerful machines in the second tier.

This topology is very flexible, providing several layers of workload management.

- The load balancer node provides IP spraying capability to multiple Web servers.
- Servlet requests can be distributed across multiple Web containers by the Web server plug-in.
- EJB requests can be distributed across multiple EJB containers using the Enterprise Java Services (EJS) workload management capability of WebSphere.

Consider the following options for clustering.

Minimal clustered

In this scenario you would cluster the business logic tier application servers, that is, the servers hosting the EJBs. The application servers in this tier act as a common service layer to the application servers in the presentation tier. As such they should be a single logical "application server" (that is, a cluster).

In this option, the application servers in the presentation tier would be non-clustered. Each would receive a request from its local Web server via the Web server plug-in, process the request in its JVM on that machine, but call the “logical” business logic layer to take advantage of EJS workload management. However, what happens if the local Web server or presentation server fail on a particular machine? In this configuration, no request processing will occur on the machine if either process fails. This is where the next, more complex, option comes into play.

Complex clustered

This option would extend the previous option by clustering the presentation tier application servers, too. The Web server plug-in on each physical server can then be configured to distribute requests across all the presentation application servers in the cluster, both local and on other machines. This way if the local application server dies, the Web server plug-in can perform its normal failover support to push the request to one of its other configured presentation application servers. Also, if the Web server process on a particular machine dies, the presentation application server JVM will still receive requests from the Web servers on the other machines.

In addition, the business logic application servers would still be clustered to take advantage of EJS workload management.

Advantages

Using an IP sprayer to distribute HTTP requests has the following advantages:

- Improved server performance by distributing incoming TCP/IP requests (in this case, HTTP requests) among a group of Web server machines.
- The use of multiple Web servers increases the number of connected users.
- Elimination of the Web server as a single point of failure. It can also be used in combination with WebSphere workload management to eliminate the application server as a single point of failure.
- Improved throughput by letting multiple servers and CPUs handle the client workload.

Disadvantages

Using an IP sprayer to distribute HTTP requests has the following disadvantage:

- Extra hardware and software are required for the IP sprayer servers.

3.3.7 Topology 6: multiple WebSphere cells

Figure 3-7 depicts how an application can be implemented over multiple cells.

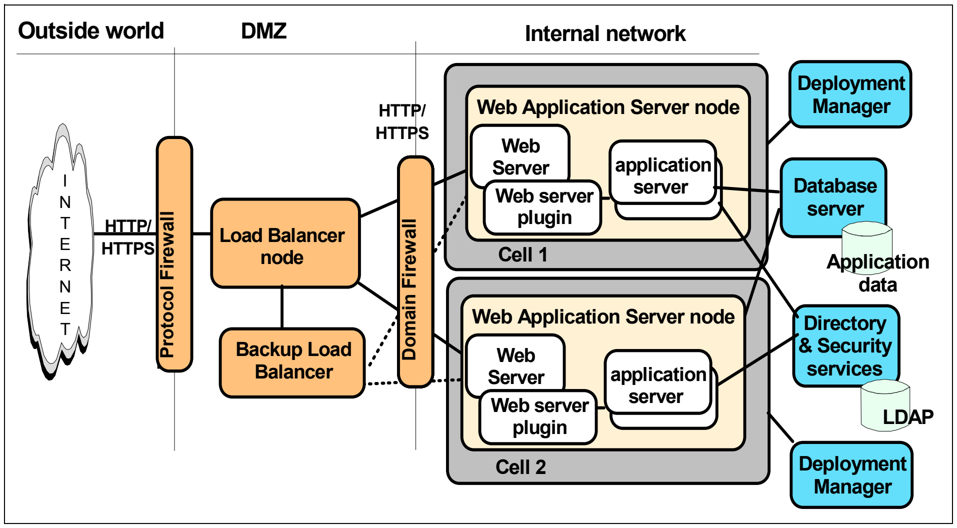


Figure 3-7 Multiple WebSphere cell topology

The example application runs simultaneously in two cells, each hosted on a different physical machine. The load balancer node is used as an IP sprayer to distribute incoming HTTP requests among the two cells, presenting a single image of the application to clients.

Each cell is administered independently, because each cell has its own set of XML configuration files. A different version of the application can be run in each cell. In addition, because the cells are isolated from one another, different versions of the WebSphere Application Server software can be used in each cell.

There are no hard limits on the number of nodes and application servers that can be used in a single cell. However, the choice of whether to use a single or multiple cells can be made by weighing the advantages and disadvantages.

Advantages

Using multiple cells has the following advantages:

- Isolation of hardware failure

  If one cell goes offline due to hardware problems, the others can still process client requests.

- Isolation of software failure

  Running an application in two or more cells isolates any problems that occur within a cell, while the other cells continue to handle client requests. This can be helpful in a variety of situations:

  – When rolling out a new application or a revision of an existing application. The new application or revision can be brought online in one cell and tested in a live situation while the other cells continue to handle client requests with the production version of the application.

  – When deploying a new version of the WebSphere Application Server software. The new version can be brought into production and tested in a live situation without interrupting service.

  – When applying fixes or patches to the WebSphere Application Server software. Each cell can be taken offline and upgraded without interrupting the application.

- If an unforeseen problem occurs with new software, using multiple cells can prevent an outage to an entire site. A rollback to a previous software version can also be accomplished more quickly. Hardware and software upgrades can be handled on a cell-by-cell basis during off-peak hours.

- Improved performance

  Running an application using multiple smaller cells may provide better performance than a single large cell because there is less inter-process communication in a smaller cell.

Disadvantages

Using multiple cells has the following disadvantages:

- Deployment is more complicated than for a single administrative cell. Using a distributed file system that provides a common file mount point can make this task easier.

- Multiple cells require more administration effort because each cell is administered independently. This problem can be reduced by using scripts to standardize and automate common administrative tasks.

Variation: Another possible multi-cell configuration exists, although it's more limited. The determining factor is that both WebSphere cells must be running the same version of WebSphere. When this is the case, you can configure a Web server and its Web server plug-in to service multiple cells simultaneously.

The configuration consists of:

- A Web server and Web server plug-in on a machine in the DMZ.
- Multiple cells and application servers configured on (and across) multiple machines behind the domain firewall.

Manual manipulation of the Web server plug-in configuration file in each cell would be required so requests are sent to application servers residing in multiple cells. Session affinity between application servers in the same cell would work, but not between application servers in multiple cells.

3.3.8 Topology 7: multiple clusters on a node

When deciding how to deploy an application, one decision to be made is whether to deploy a cluster of application servers across all machines, as depicted in Figure 3-8, or only on a single machine, as depicted in Figure 3-9 on page 112.

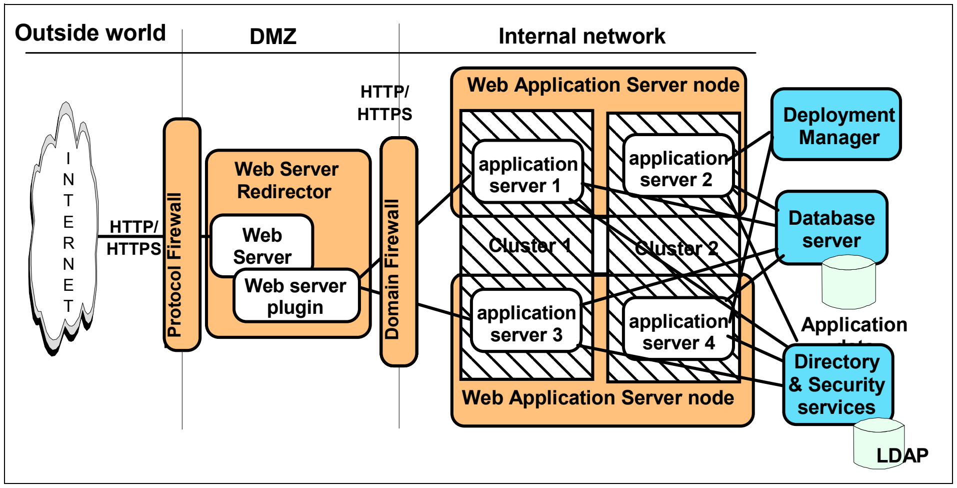


Figure 3-8 Multiple application server clusters on each machine

The topology in Figure 3-8 uses horizontal scaling. Each cluster of application servers is distributed throughout all of the machines in the system. In this example, a member of each cluster is hosted on each Web application server node. The Web server plug-in distributes client requests to the application servers on each node.

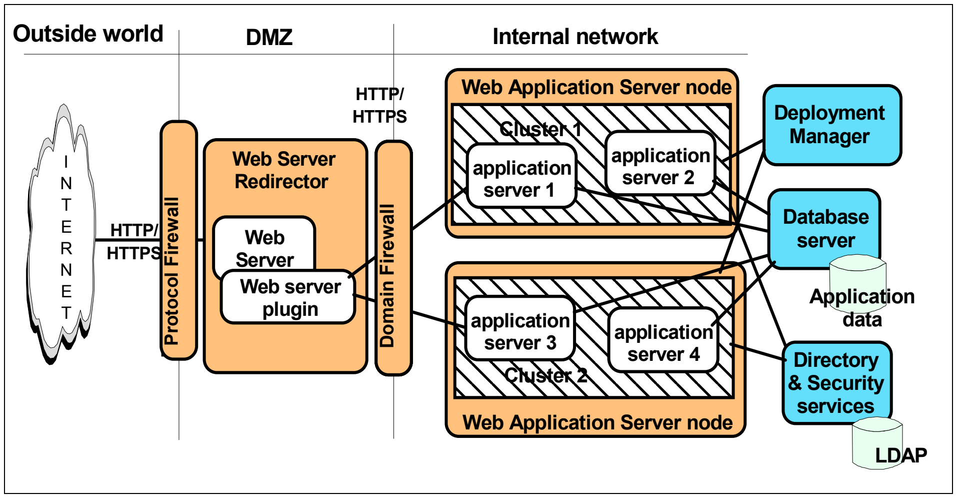


Figure 3-9 Single application server cluster on each machine

The topology in Figure 3-9 uses vertical scaling. Each cluster of application servers is located on a single machine of the system.

Advantages

Hosting members of multiple application server clusters across one or more machines has the following advantages:

- Improved throughput

  Using an application server cluster enables the handling of more client requests simultaneously.

- Improved performance

  Hosting cluster members on multiple machines enables each member to make use of the machine's processing resources.

- Hardware failover

  Hosting cluster members on multiple nodes isolates hardware failures and provides failover support. Client requests can be redirected to the application server members on other nodes if a node goes offline.

- Application software failover

  Hosting cluster members on multiple nodes also isolates application software failures and provides failover support if an application server stops running. Client requests can be redirected to cluster members on other nodes.

- Process isolation

  If one application server process fails, the cluster members on the other nodes are unaffected.

Disadvantages

Hosting members of multiple application server clusters across one or more machines has the following disadvantage:

- More complex maintenance

  Application servers must be maintained on multiple machines.

3.3.9 Topology 8: combined topology

An example of a topology that combines the best elements of the other topologies discussed in this chapter is depicted in Figure 3-10 on page 114.

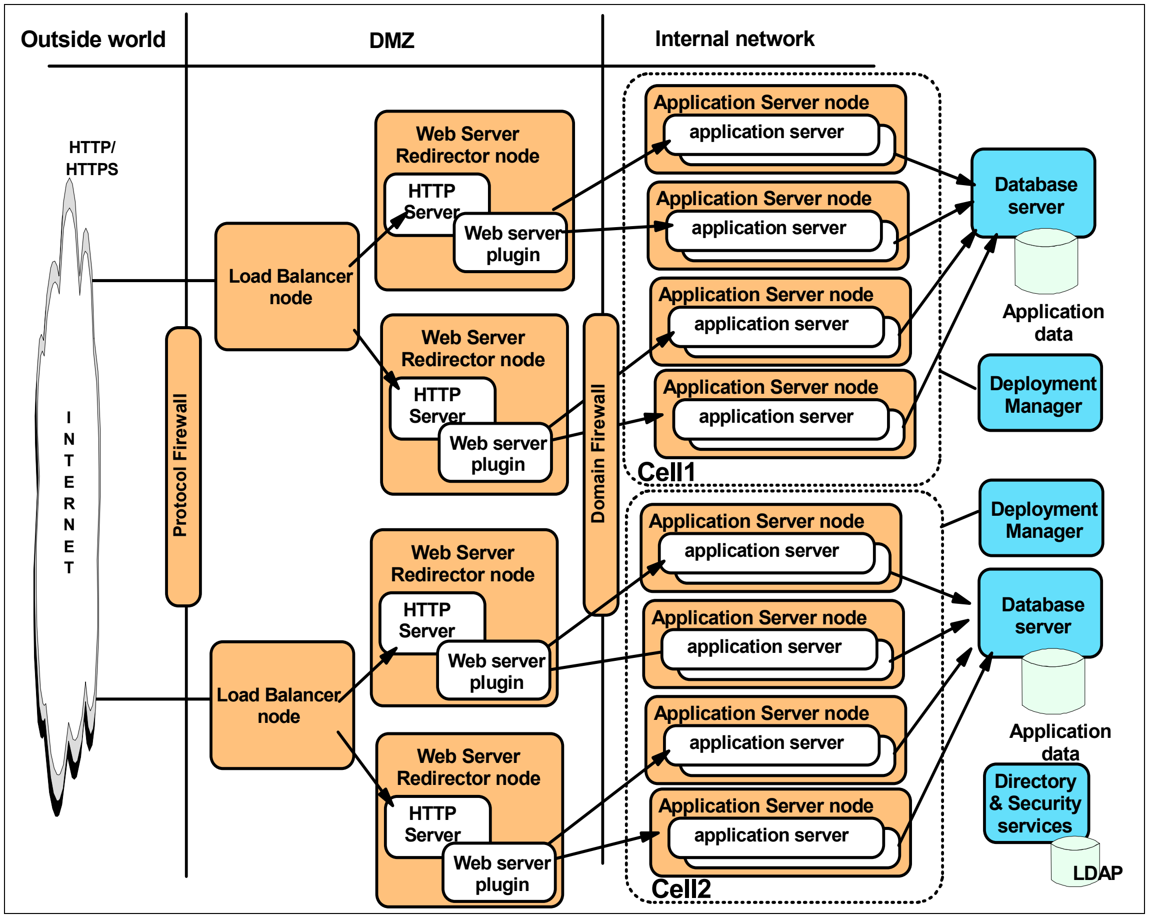


Figure 3-10 Combined topology

This topology combines elements of several different basic topologies:

- Two WebSphere cells.
- Two load balancer nodes.
- Two HTTP servers for each cell with the Web server plug-in.
- Four application server machines for each cell.
- The use of application server clusters for both vertical and horizontal scaling. In the example topology, each machine hosts two cluster members; in practice, the number of cluster members is limited by the computing resources of each node.
- Two database servers for each cell. These servers host mirrored copies of the application database.

Advantages

This topology is designed to maximize throughput, availability, and performance. It incorporates the best practices of the other topologies discussed in this chapter:

- Having more than one load balancer node, HTTP server, application server, and database server in each cell eliminates single points of failure.
- Multiple cells provide both hardware and software failure isolation, especially when upgrades of the application or the application server software are rolled out. (Hardware and software upgrades can be handled on a cell-by-cell basis during off-peak hours.)
- Horizontal scaling is done by using both application server clusters and the IP sprayer to maximize availability and eliminate single points of process and hardware failure.
- Application performance is improved by using several techniques:
  - Hosting application servers on multiple physical machines to boost the available processing power.
  - Creating multiple smaller cells instead of a single large cell. There is less interprocess communication in a smaller cell, which allows more resources to be devoted to processing client requests.
  - Using clusters to vertically scale application servers on each node, which makes more efficient use of the resources of each machine.
- Applications with this topology can make use of several workload management techniques. In this example, workload management can be performed through one or more of the following:
  - Using the IBM WebSphere Application Server Network Deployment workload management facility to distribute work among clustered application servers.
  - Using the load balancer to distribute client HTTP requests to each HTTP server.

  For example, an application can manage workloads at the HTTP server level with the load balancer and at the application server level with WebSphere workload management. Using multiple workload management techniques in an application provides finer control of load balancing.

  Regardless of which workload management techniques are used in the application, Network Deployment participates in workload management to provide failover support.
- In this topology, users notice an interruption only when an entire cell is lost. If this occurs, the active HTTP sessions are lost for half of the clients. The

system can still process HTTP requests, although its performance is degraded.

Disadvantages

The combined topology has the following disadvantage:

- Multiple cells require more administration effort, because each cell is administered independently. This problem can be reduced by using scripting to standardize and automate common administrative tasks.

3.4 Closing thoughts on topologies

Whatever topology is decided upon, a best practice is to partition the production acceptance environment exactly the same as the production environment. This avoids surprises when deploying an application into production.

Another consideration, when practical for an application architecture, is to create a number of smaller application servers, rather than a single large server. This has two advantages:

- The Web server plug-in configuration files can be smaller (less complexity of URIs), which leads to better startup performance and possibly better execution performance.
- At least during the development phase, it takes less time to cycle a smaller application server to pick up various configuration changes.

Of course, creation of multiple application servers in this manner needs to be carefully balanced against the increased complexity of doing so and the potential increase in response time due to inter-process RMI/IIOP calls and network latency.

3.5 For more information

- *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198
- *Patterns for the Edge of Network*, SG24-6822

4

Installation approach

This chapter provides a summary of the decisions that must be made when planning an installation to match a particular topology or architecture. Once you have read the information here, you can find more detailed information about the installation process in the following:

- Chapter 5, “Windows 2000 installation steps” on page 139
- Chapter 6, “AIX installation steps” on page 167
- *IBM WebSphere V5.0 for Linux, Implementation and Deployment Guide*, REDP3601
- *WebSphere Installation, Configuration, and Administration in an iSeries Environment*, SG24-6588

This chapter concentrates on the decisions and tasks associated with a “clean” installation. If you are migrating, the following IBM Redbooks are available to help you:

- For migration from WebSphere Application Server V3.5 or V4.0, see *Migrating to WebSphere V5.0: An End-to-End Migration Guide,* SG24-6910.
- For migration from BEA WebLogic, see *Migrating WebLogic Applications to WebSphere V5,* REDP0448.
- For coexistence and other detailed installation information, refer to the *IBM WebSphere Application Server, Version 5 Getting Started* book. It is available in both HTML and PDF form through the InfoCenter.

4.1 Selecting a topology

The first step in planning your installation is to use the information in Chapter 3, "Topology selection" on page 85 to determine the layout of your network. The decisions you make while determining the system layout will affect your installation decisions.

4.1.1 Using the embedded HTTP transport as opposed to a stand-alone Web server

In topologies that show the Web server on a separate physical node from the application server, a Web server plug-in is required to take incoming requests and transport them to the appropriate Web container. However, some topologies show the Web server and the application server on the same machine. In those instances it is not mandatory that you install and configure a separate Web server.

Each Web container in WebSphere Application Server V5 includes an embedded HTTP transport. Although its primary purpose is to service requests from the Web server plug-in, the transport can also be accessed directly by any HTTP client. This approach is not recommended for a production environment. However in a test environment, using the embedded HTTP transport has the following advantages:

- All application changes are immediately available to the transport as a result of the direct access to the administrative database. There is no Web server to configure or Web server plug-in configuration file to regenerate whenever a servlet URL is added or changed.
- The WebSphere Application Server can be verified by directly accessing test or application URLs without having to make the request via an intervening Web server. Any problems can be immediately identified as resulting from the WebSphere configuration, not Web server or Web server plug-in problems.
- By requesting an application's URLs directly through the embedded HTTP transport, an application can be verified prior to configuring a Web server plug-in to support the application.

In a production environment we recommend that you use a stand-alone Web server and the Web server plug-in. These can reside on the same machine as the WebSphere Application Server but a better approach is to put these on a separate machine, preferably in a secure DMZ.

The Web server plug-in uses an XML configuration file (plugin-cfg.xml) containing settings that describe how to handle and pass on requests to the

WebSphere Application Server(s) made accessible through the Web server plug-in.

Note: The OSE plug-in provided in previous releases of WebSphere Application Server is not supported in WebSphere V4 or in WebSphere V5. All installations should now use the new Web server plug-in.

Using a stand-alone Web server and the Web server plug-in offers the following advantages:

- Workload management:

  The Web server plug-in can balance requests among Web containers. The embedded HTTP transport does not support workload management.

- Performance:

  Web servers are designed for serving static content . When using the embedded transport, all static content, as well as dynamic, must be served by the WebSphere Application Server.

- Security:

  Using a stand-alone Web server and the Web server plug-in allows you to separate the Web server and WebSphere interfaces, allowing you to place the Web server in a DMZ and the WebSphere Application Server behind a firewall.

- Availability:

  You can size and configure servers appropriate to each task. By separating the Web server from the WebSphere Application Server, each machine can be sized and configured to optimize the performance of each component. In addition, a component's server can be reconfigured, or even replaced, without affecting the installation of the other component.

- Remove resource contention:

  By installing the Web server on a separate machine from the WebSphere Application Server, a high load of static requests will not affect the resources (CPU, memory, disk) available to WebSphere, and therefore will not affect its ability to service dynamic requests.

4.2 Packaging

The WebSphere Application Server 5.1 base package comes with the following installable images (this may vary depending on the platform):

- WebSphere Application Server
- IBM HTTP Server
- Web server plug-ins
- Application Client
- Application Server Toolkit
- DataDirect JDBC Driver

The WebSphere Application Server Network Deployment package contains the same contents as the base package, plus the following:

- Deployment Manager
- Edge Components
- IBM DB2-Enterprise Edition
- IBM Directory Server

4.2.1 Determining which features to install

Most topologies will fall into one of the scenarios in Table 4-1. You can use this table to determine which products you will need to install and on which machines.

Table 4-1 Installation overview

Topologies	Server 1	Server 2	Server 3(+)	Server 4(+)	Server 5(+)
► Test environment	► IHS ► WAS ► plug-in				
Legend: ► WAS: WebSphere Application Server ► IHS: IBM HTTP Server or other supported Web server package ► Dmgr: Deployment Manager ► Edge: Edge Components Load Balancer ► plug-in: Web server plug-in ► LDAP: IBM Directory Server 4.1					

Topologies	Server 1	Server 2	Server 3(+)	Server 4(+)	Server 5(+)
▸ "Topology 1: vertical scaling" on page 98 ▸ "Topology 2: HTTP server separation" on page 99 ▸ "Topology 4: horizontal scaling with clusters" on page 104 ▸ "Topology 7: multiple clusters on a node" on page 111	▸ IHS ▸ plug-in	▸ Dmgr	▸ LDAP	▸ WAS	
▸ "Topology 3: reverse proxy" on page 102	▸ Edge	▸ IHS ▸ plug-in ▸ WAS	▸ LDAP		
▸ Topology 5: "Simple IP sprayer topology" on page 106 ▸ "Topology 6: multiple WebSphere cells" on page 109	▸ Edge	▸ Dmgr	▸ LDAP	▸ IHS ▸ plug-in ▸ WAS	
▸ Topology 5: "Complex IP sprayer topology" on page 107	▸ Edge	▸ Dmgr	▸ LDAP	▸ IHS ▸ plug-in ▸ WAS	▸ WAS
Legend: ▸ WAS: WebSphere Application Server ▸ IHS: IBM HTTP Server or other supported Web server package ▸ Dmgr: Deployment Manager ▸ Edge: Edge Components Load Balancer ▸ plug-in: Web server plug-in ▸ LDAP: IBM Directory Server 4.1					

General note: In addition to the packages shown, it is most likely that you will want to install a relational database product for application and persistence data. The database product can be installed on a separate server or on one of the application or Network Deployment servers.

4.3 Selecting the platform and checking the requirements

The choice of operating system for each component will depend on your budget, skills, performance requirements, and other factors. In this section, we will not try to lead you in one direction or another.

At the time of writing this redbook, WebSphere Application Server 5.1 supports the following operating systems:

- AIX
- OS/400
- Connectiva Linux Enterprise Edition
- RedHat Enterprise
- SuSE SLES
- TurboLinux Enterprise Server
- United Linux
- Solaris
- HP-UX
- Windows 2000 Server and Advanced Server
- Windows 2003 Standard and Enterprise

For the most current information on the supported software releases, operating systems, and maintenance levels, see:

`http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

The installation process will also check for the presence of the prerequisites and will alert you if the requirements are not met. You will have a choice to continue with the install or to cancel.

4.4 Silent versus GUI installation

The WebSphere Application Server installation program provides two modes of operation:

- GUI installation
- Silent installation

Note: The native installation method provided in previous versions of WebSphere Application Server is no longer available.

GUI installation

An interactive installation displays a graphical user interface (GUI) within which the choices are displayed a window at a time, allowing easy setup and execution of the installation.

The GUI installation option should be used for:

- First time or inexperienced users.
- One-time installs.

- Determination of the settings required to create a script file used for later silent installations.
- Testing of installations involving different combinations of software components.

Silent installation

A silent installation is a non-interactive installation that reads all choices from a script file that must be pre-prepared by the user.

The silent installation option should be used for:

- Installation of identical configurations on multiple machines.
- Backup of the installation settings for later reuse.
- Scripted or unattended installations.
- Duplication of development or testing installation in production.

If you choose silent installation, you will need to modify a response file to be used during the installation.

4.5 Planning for IBM HTTP Server

The use of a Web server is recommended for all topologies. In addition to the Web server, you will need to install the Web server plug-in on the same server.

At the time of writing this redbook, the following Web servers were supported:

- Apache Server
- IBM HTTP Server
- Internet Information Server
- Sun ONE Web Server (formerly iPlanet), Enterprise Edition
- Lotus Domino Enterprise Server

This installation scenarios in this book assume that you will be installing the IBM HTTP Server and the corresponding Web server plug-in from the WebSphere Application Server 5.1 installation CD.

4.5.1 Checking for IP port conflicts

The IBM HTTP Server will use three IP ports on the server. Check that there are no existing services on the server that use these ports:

Table 4-2 IP ports used by IBM HTTP Server

Port	Use
80	Standard HTTP port
443	Standard HTTPs port
8008	Administrative console

The following commands can be used to see a list of ports in use:

- Windows: **`netstat -an`**
- AIX, Linux, and Solaris: **`netstat -an | grep LISTEN`**

If your operating system has a pre-installed Web server and you have decided to install the IBM HTTP Server, you will need to remove the existing Web server or prevent it from starting to avoid port conflicts.

4.5.2 Run as a Windows service

If you are installing on a Windows platform, you will have the option to run as a service. If you choose to do this, the installation will add the following services:

- IBM HTTP Administration 1.3.28
- IBM HTTP Server 1.3.28

Both will be set to start automatically.

A user ID and password are required to run the service. The user ID you specify will be given the following user rights during the installation:

- Act as part of the operating system
- Log on as a service

4.5.3 Installation and configuration

The steps required to install and configure the IBM HTTP Server to work with a WebSphere Application Server system are:

1. Install the IBM HTTP Server and the Web server plug-in. Both are included on the same CD as WebSphere Application Server 5.1. You can install them simultaneously by using the custom install path of the WebSphere installation.
2. The IBM HTTP Server can be administered by manually updating its configuration file or by using the IBM HTTP Server administration interface. If you choose to use the GUI interface, you will need to specify an administrator user ID and password for the IBM HTTP Server configuration.

3. Verify the installation and the administrator interface.
4. (Optional) Enable SSL. For information about enabling SSL security between the Web server and the Web client, and between the Web server plug-in and WebSphere, see the *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

Note: The Web server plug-in needs to be generated before it will work with a WebSphere Application Server installation. This generation is done from the administrative console after WebSphere has been installed. The steps required are:

1. Generate the Web server plug-in. Before the IBM HTTP Server can forward requests to WebSphere, the Web server plug-in must be generated from the WebSphere Application Server system.
2. Copy the plug-in configuration file to the IBM HTTP Server file system.
3. Verify that the plug-in is working correctly.

These steps are addressed as post-installation tasks for WebSphere Application Server.

4.6 Planning for WebSphere Application Server

During the installation of WebSphere Application Server V5 you will have to make several decisions. This section gives you some guidance on making these decisions.

4.6.1 Selecting the features to install

The WebSphere Application Server installation offers two installation options:

- **Full install:** The full install includes all features. This is primarily recommended for a test environment. In a production environment you will want to spread out the features on multiple machines. For example, the HTTP server and Web server plug-in should be installed on a machine separate from WebSphere Application Server.
- **Custom install:** Allows you to select product features. We recommend that the custom installation option be used, since it provides the choices and flexibility needs for all but the simplest of WebSphere Application Server configurations.

Table 4-3 on page 126 lists the options you can select from for a custom installation.

Table 4-3 WebSphere Application Server installation feature options

Feature	**Test application servers**	**Production application servers**	**Web servers**	**Other**
Application Server	X	X		
Application Server samples	X			
Scripted Administration	X	X		
Administrative console	X[1]	X		
Ant and Deployment Tools	X			Development machines
Embedded messaging features	X	X[2]		
Message Driven Bean Samples	X			
IBM HTTP Server	X		X	
Web server plug-ins	X		X	
Tivoli Performance Viewer	X			Any machine used to monitor
Dynamic Cache Monitor	X	X		
Performance Servlet	X	X		
Log Analyzer	X	X		
Javadocs	X	X		

[1] The administrative console gets removed when a server is added to a cell.
2 The embedded messaging features are necessary if you plan to use the embedded WebSphere JMS provider.

Important: Although not listed in the custom install window, the IBM JDK 1.4.1 is automatically installed in the <WAS_HOME>/java directory. There is no need to separately install a JDK for use by WebSphere Application Server.

4.6.2 Checking for IP port conflicts

By default, WebSphere Application Server V5 uses the following IP ports.

Table 4-4 IP ports used by WebSphere Application server

Port	Use
9080	HTTP transport
9443	HTTPS transport
9090	Administrative console
9043	Administrative console secure port
2809	Bootstrap
8880	SOAP connector
7873	DRS client
5558	JMS server queued
5557	JMS server security
5559	JMS server direct
9101	ORB listener

The following commands can be used to see a list of ports in use:

- Windows: **`netstat -an`**
- AIX, Linux, and Solaris: **`netstat -an | grep LISTEN`**

If you have conflicts with existing applications and services, you can change these during the installation.

AIX users: Port 9090 might already be used by the Web-based system manager of AIX V5.1. In this case, the port number for the WebSphere administrative console should be changed after the installation is complete.

4.6.3 Run as a service (Windows only)

If you elect to run WebSphere as a service, you will need to supply a user ID for it to run under. The user ID must exist and must have the following rights:

- Act as part of the operating system
- Log on as a service

If the user ID you specify does not already have these rights assigned, they will be given during the installation, but a reboot will be required for these authorities to take place. You can assign rights to a user by selecting **Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> User Rights Assignment**.

The following services will be added:

- IBM WebSphere Application Server - server1
- WebSphere Embedded Messaging Publish and SubscribeWAS_<node>_server1 (if messaging features were installed)

The initial setting for these services is for a manual start.

4.6.4 Embedded messaging considerations (UNIX systems)

If you want to install the embedded messaging features, the JMS server will require new users and groups to be defined to the operating system. If you are installing on a Windows system, nothing is required. The installation will create the required users and groups.

On UNIX systems, you will need to create two groups and one user, and add the user ID that WebSphere will run under to these groups.

1. Log in as root and start a terminal session.
2. Create two new groups called mqm and mqbrkrs.
3. Create a user called mqm and add it to the mqm group.
4. Add the WebSphere user ID (that is, root) to both groups.
5. Log off and then on again to set the permissions.

For embedded messaging to work correctly on Windows, the account under which the WebSphere process is run must have an account name of 12 characters or less (an MQ limitation). So for example, don't run WebSphere Application Server V5 under the Administrator account (13 characters). Use something like “wasadmin” instead. This goes both for running WebSphere Application Server as a service and for starting WebSphere Application Server processes from the command line with `Start` commands.

4.6.5 Installation and post-installation tasks

The installation of WebSphere Application Server, whether done using the GUI interface or a silent install is a straightforward task. The post installation tasks required for minimum setup will depend largely on the configuration of your environment.

Network Deployment environment: If you are installing WebSphere Application Server with the intention of adding it to a cell, you do not need to do these post-installation tasks, with the exception of running the installation verification procedures.

The basic steps for installation and post-install tasks are:

- Install WebSphere Application Server.
- Run the installation verification procedures.
- Start the server.
- Verify the default and sample applications are working.
- Install applications.
- Configure the Web server plug-in and verify it is working correctly.
- Implement security.

4.7 Planning for Network Deployment

This section will focus on installing the WebSphere Deployment Manager from the Network Deployment package.

Edge Components: This book does not go into any details about the Edge Components. The features shipped Edge Components include the Load Balancer and Caching Proxy components of WebSphere Edge Server. For further information about the Edge Components, see *WebSphere Edge Server New Features and Functions in Version 2*, SG24-6511 and *Patterns for the Edge of Network*, SG24-6822.

4.7.1 Selecting the features to install

When you install the Deployment Manager, you can select the full install or custom install. A full install will install all of the features. If you choose custom, you can select from the following features:

- Deployment Manager: you will need to install one Deployment Manager per cell.
- Embedded messaging client: If you are using the embedded WebSphere JMS provider for messaging, you should install this. For information about the messaging options, see Chapter 11, “Asynchronous messaging” on page 451.
- UDDI Registry, Web Services Gateway: The installation and use of these features are beyond the scope of this book. If you will be implementing the UDDI Registry or Web Services Gateway, see *WebSphere Version 5 Web Services Handbook,* SG24-6891. These are advanced features and if you don’t know you need them, skip them for now.

Important: Although not listed in the custom install window, the IBM JDK 1.4.1 is automatically installed in the <WAS_ND_HOME>/java directory. There is no need to separately install a JDK for used by WebSphere Application Server Network Deployment.

4.7.2 Checking for IP port conflicts

By default, WebSphere Application Server Network Deployment uses the ports shown in Table 4-5.

Table 4-5 IP ports used by WebSphere Application server

Port	Use
9090	administrative console[1]
9043	administrative console secure port[1]
9809	Bootstrap
8879	SOAP connector
7989	DRS client

[1] This port is the same as used by a base WebSphere Application Server installation. If you are installing Network Deployment on a system that has an existing base installation, you will need to select a different port *unless* you are going to immediately add that base installation to the Deployment Manager cell. In that case, the base admin console application is deleted and there is no port conflict.

Port	Use
9401	SAS SSL ServerAuth Address
9403	CSIV2 ServerAuth Listener Address
9402	CSIV2 MultiAuth Listener Address
9100	ORB listener
7277	Cell Discovery Address

[1] This port is the same as used by a base WebSphere Application Server installation. If you are installing Network Deployment on a system that has an existing base installation, you will need to select a different port *unless* you are going to immediately add that base installation to the Deployment Manager cell. In that case, the base admin console application is deleted and there is no port conflict.

The following commands can be used to see a list of ports in use:

- Windows: **`netstat -an`**
- AIX, Linux, and Solaris: **`netstat -an | grep LISTEN`**

If you have conflicts with existing applications and services, you can change these during the installation.

AIX users: Port 9090 might already be used by the Web-based system manager of AIX V5.1. In this case, the port number for the WebSphere administrative console should be changed after the installation is complete.

4.7.3 Run as a service (Windows)

If you elect to run WebSphere as a service, you will need to supply a user ID for it to run under. The user ID must have the following rights:

- Act as part of the operating system
- Log on as a service

If the user ID you specify does not already have these rights assigned, they will be given during the installation, but a reboot will be required for these authorities to take place. You can assign rights to a user by selecting **Control Panel -> Administrative Tools-> Local Security Policy -> Local Policies -> User Rights Assignment**.

The following service will be added:

- IBM WebSphere Application Server V5 - dmgr

The initial setting for this service is for a manual start.

4.7.4 Installation and post-installation tasks

After installing WebSphere Application Server Network Deployment, there will be some basic post-installation tasks that need to be performed to set up a minimum operating environment:

- Start the Deployment Manager.
- Start the administrative console.
- (Optional) Secure the administrative console. When you install the Deployment Manager, the console installed with the Deployment Manager becomes the single console for the cell. As you add WebSphere Application Server base instances to the cell, the individual console for each application server is removed and the configuration is added to the Deployment Manager configuration repository. The process for securing the console is discussed in 8.5, "Securing the administrative console" on page 276.
- Add nodes to the cell. Immediately after installation, you have a cell and a Deployment Manager but you do not have any application servers in the cell.
- (Optional) Define clusters.
- Install user applications.

  If you installed the WebSphere samples, they are already deployed and ready to test. To see the samples, open a Web browser and go to:

  ```
  http://<hostname>/WSsamples
  ```

4.8 Planning for and installing messaging products

In J2EE 1.3, JMS becomes an integral part of the J2EE platform. WebSphere Application Server V5 satisfies this requirement by including a JMS provider as part of the installation. You can use the internal JMS provider, or you may choose to use an external provider. WebSphere supports the following JMS providers:

- WebSphere JMS provider

  The WebSphere JMS provider is installed (embedded) as part of WebSphere Application Server. The implementation is based on a reduced footprint of the following products: IBM's WebSphere MQ product, the WebSphere MQ Event Broker product and the JMS Client ("MA88") support pack.
- WebSphere MQ JMS provider

  WebSphere Application Server V5 supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

- Generic JMS providers

  WebSphere Application Server V5 supports the use of generic JMS providers, as long as they implement the ASF component of the JMS 1.0.2 specification.

4.8.1 Installing and verifying the WebSphere JMS provider

The embedded WebSphere JMS provider gets installed as part of the installation of WebSphere Application Server on a node.

Note: If you perform a custom installation, make sure that **Embedded Messaging**, **Server and Client** is selected among the features you would like to install.

Verifying the installation

Check the following to verify that the embedded messaging features have installed correctly on a node:

1. Check <WAS_HOME>/logs/mq_install.log

   This log should have a number of sections, including:

   – Prerequisites checking
   – WebSphere MQ install log
   – JMS install log
   – Publish/Subscribe install log

   All of the sections should end with something that looks like ERROR_SUCCESS (0).

2. Check the <WAS_HOME>/logs/createMQ.log file on the node.

   The install process should have created the Base queue manager and a number of MQ default queues. The queue manager will have a name of the form:

   WAS_<NodeName>_server1

   For example, for a base application server installation the queue manager will be located at:

   <mq_home>/Qmgrs/WAS_<NodeName>_server1

   When the node is added into a cell, a new queue manager is created to replace the base queue manager:

   WAS_<NodeName>_jmsserver

3. On Windows, you can also see the two components that make up Embedded Messaging by looking in Add/Remove Programs in the Control Panel:

- IBM WebSphere MQ for point-to-point messaging.
- IBM WebSphere Embedded Messaging Publish And Subscribe Edition (EMPS) for publish/subscribe broker.

Starting the JMS Server

During installation, a JMS Server managed process is automatically created for the node in which the WebSphere JMS provider is installed. In a base WebSphere installation, the JMS Server runs in the same JVM as the application server (that is, it is part of the application server process). However, when the node gets added into a cell, a WebSphere Application Server managed server jmsserver that runs in its own JVM, is created. This JVM is managed by the node agent.

The JMS Server process on a node provides the JMS functions of the JMS Server on that node. It is responsible for managing the queue manager (stop and start) and hosts the broker.

Note: When using the WebSphere JMS provider, there is no WebSphere administrative console support for queue manager or channel administration.

Messaging queues for the WebSphere JMS provider (as opposed to administrative queue objects) also get created when we define them to a node's JMS Server process.

Note: There is no WebSphere administrative console support for administering the messaging queues, for example sending test messages or clearing messages from a queue.

In a Network Deployment environment, make sure that the node agent and the JMS server have been started in order for the embedded messaging service to function properly. To start the node agent and server, enter the following:

```
cd <WAS_HOME>/bin
startnode
startserver jmssserver
```

Look in the JMS Server's SystemOut.log file located on the node at:

<WAS_HOME>/logs/jmsserver/SystemOut.log

The log will contain entries similar to those in Example 4-1 on page 135:

Example 4-1 jmsserver's SystemOut.log file in the node

```
[8/14/02 17:45:36:453 EDT] 50d39cf0 JMSProvider   A MSGS0050I: Starting the
Queue Manager
[8/14/02 17:45:40:688 EDT] 50d39cf0 JMSProvider   A MSGS0051I: Queue Manager
open for business
[8/14/02 17:45:40:719 EDT] 50d39cf0 JMSProvider   A MSGS0052I: Starting the
Broker
[8/14/02 17:45:46:500 EDT] 50d39cf0 JMSProvider   A MSGS0053I: Broker open for
business
```

Notice in Figure 4-1 that when the JMS Server process is started, a number of additional processes (amq*) are started. These are all part of the WebSphere MQ queue manager. In addition there are two runmq* processes. These processes are created when the JMS Server process starts the queue manager for the JMS provider.

For more information on these processes, refer to the *MQSeries System Administration Guide*, SC33-1873-02.

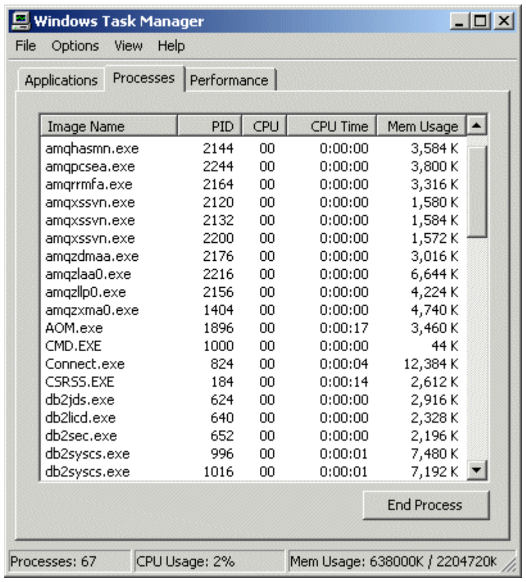


Image Name	PID	CPU	CPU Time	Mem Usage
amqhasmn.exe	2144	00	0:00:00	3,584 K
amqpcsea.exe	2244	00	0:00:00	3,800 K
amqrrmfa.exe	2164	00	0:00:00	3,316 K
amqxssvn.exe	2120	00	0:00:00	1,580 K
amqxssvn.exe	2132	00	0:00:00	1,584 K
amqxssvn.exe	2200	00	0:00:00	1,572 K
amqzdmaa.exe	2176	00	0:00:00	3,016 K
amqzlaa0.exe	2216	00	0:00:00	6,644 K
amqzllp0.exe	2156	00	0:00:00	4,224 K
amqzxma0.exe	1404	00	0:00:00	4,740 K
AOM.exe	1896	00	0:00:17	3,460 K
CMD.EXE	1000	00	0:00:00	44 K
Connect.exe	824	00	0:00:04	12,384 K
CSRSS.EXE	184	00	0:00:14	2,612 K
db2jds.exe	624	00	0:00:00	2,916 K
db2licd.exe	640	00	0:00:00	2,328 K
db2sec.exe	652	00	0:00:00	2,196 K
db2syscs.exe	996	00	0:00:01	7,480 K
db2syscs.exe	1016	00	0:00:01	7,192 K

Processes: 67 CPU Usage: 2% Mem Usage: 638000K / 2204720K

Figure 4-1 WebSphere MQ processes started by the JMS Server process

Note: There can be at most one JMS server process on each node in the cell. Any of the application servers within the cell can access the resources (queues) associated with any of the JMS Server processes in the cell.

4.8.2 Installing WebSphere MQ as the JMS provider

To install WebSphere MQ with support for JMS, complete the following steps:

1. Check the following Web site for supported levels of WebSphere MQ:

   `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

   Install a supported version of WebSphere MQ server and CSD. For this publication we used WebSphere MQ 5.2.1 with CSD5. The currently listed version supported is WebSphere MQ 5.3 with CSD 1.

   Do not install the WebSphere MQ classes for Java or JMS from the shipped CD.

2. Download and install the WebSphere MQ 5.2 MA88 SupportPac (WebSphere MQ classes for Java and WebSphere MQ classes for Java Message Service) for your operating system, found at:

   `http://www-4.ibm.com/software/ts/mqseries/txppacs/ma88.html`

   For information about how to install the MA88 SupportPac, see the installation instructions for your operating system provided on the download Web page.

3. If you want to use WebSphere MQ Publish/Subscribe support, you need to provide a Publish/Subscribe broker. At present, you can do this by using either MQ Publish and Subscribe, WebSphere MQ Integrator, or the WebSphere MQ Publish/Subscribe MA0C SupportPac.

   For more information about WebSphere MQ Integrator, see the WebSphere MQ Integrator Web site at:

   `http://www-4.ibm.com/software/ts/mqseries/platforms/#integrator`

   If you want to use the MA0C SupportPac, complete the following substeps:

   a. Download the WebSphere MQ 5.2 MA0C SupportPac (WebSphere MQ -Publish/Subscribe) for your operating system from the following URL:

   `http://www-4.ibm.com/software/ts/mqseries/txppacs/ma0c.html`

   b. Install the MA0C SupportPac as described in the installation instructions for your operating system provided on the download Web page.

   Note: The use of MA0C as your publish/subscribe broker is now discouraged. However, it does work and is widely available. You should use WebSphere MQ Event Broker 2.1 (when available) or MQ Integrator instead.

4. Follow the WebSphere MQ 5.2 instructions to verify your installation setup.

5. For UNIX platforms, add the following path to your LD_LIBRARY_PATH environment variable: <mq_home>/mqm/java/lib

7. For UNIX platforms, copy the following files from the <mq_home>/mqm/java/lib directory to the <WAS_HOME>/lib/ext directory:

   – fscontext.jar
   – providerutil.jar
   – com.ibm.mq.jar
   – com.ibm.mqjms.jar

8. For UNIX platforms, copy the following files from the <mq_home>/mqm/java/lib directory to the <WAS_HOME>/bin directory:

   – libmqjbnd02.so
   – libMQXAi01.so

The full version of WebSphere MQ can be installed after installing the embedded WebSphere JMS provider. However, you will end up with only one copy of the message transport.

Existing JMS resource definitions for the WebSphere JMS provider will continue to work with WebSphere MQ as the JMS provider, so you do not need to redefine those JMS resources.

Make sure that WebSphere MQ and the JMS Client are at the required levels for embedded messaging. You won't be able to install a version of WebSphere MQ that is lower than the one currently installed.

Note: Complete the following steps to install the full WebSphere MQ Provider after installing the WebSphere JMS provider:

1. Install a recommended version of WebSphere MQ server. Run **`setup.exe`**.
2. Install any recommended CSDs .
3. When asked if you want to remove or modify the version of WebSphere MQ currently installed, choose to remove the **Server**.
4. Select to keep existing queue managers.
5. Proceed with the installation of the full WebSphere MQ Server by running **`setup.exe`** again, after the old WebSphere MQ server has been removed.
6. Don't install the Client. Use the Client that was installed with the WebSphere JMS provider.

It is also recommended that you use WebSphere MQ Event Broker or MQSI as your publish/subscribe broker instead of MA0C.

4.8.3 Installing and defining a generic JMS provider

If you want to use a JMS provider other than the embedded WebSphere JMS provider or a WebSphere MQ JMS provider, you should complete the following steps:

1. Install and configure the JMS provider.
2. Create and configure its resources by using the tools and information provided with the JMS provider.
3. Define a generic JMS provider in WebSphere using the **Resources -> Generic JMS Providers** menu option in the administrative console.
4. Bind resources into the application server’s name space.

5

Windows 2000 installation steps

This chapter provides detailed procedures for installing, configuring, and verifying IBM HTTP Server, WebSphere Application Server, and WebSphere Network Deployment on Windows 2000. Before beginning your installation, read Chapter 4, “Installation approach” on page 117. It will lead you through the planning process necessary for a successful install. The topics in this chapter are:

- Installation of the IBM HTTP Server and/or the Web server plug-in
- Installation of WebSphere Application Server
- Installation of Network Deployment

The intent of this chapter is to lead you through a simple, first-time installation. It should help you understand the basic process of installing a base or Network Deployment environment, accessing the default and sample applications, and accessing the administration facilities.

If you have any issues with coexistence or migration, please refer to the Installation Guide provided with the product.

5.1 Product installation root variables

The variables listed in Table 5-1 are used frequently throughout this documentation to represent the root installation directories of the software components.

Table 5-1 Product Installation roots

Variable	Default value	Component
<WAS_HOME>	c:\Program Files\WebSphere\AppServer	WebSphere Application Server
<WAS_ND_HOME>	c:\Program Files\WebSphere\DeploymentManager	WebSphere Application Server Network Deployment
<IHS_HOME>	c:\Program Files\IBMHttpServer	IBM HTTP Server
<PLUGIN_HOME>	c:\Program Files\WebSphere\AppServer	Web server plug-in

5.2 Installing the IBM HTTP Server and/or Web server plug-in

This section provides instructions for installing, configuring, and verifying IBM HTTP Server V1.3.28 and the Web server plug-in for Windows 2000. These instructions also apply if you have an existing Web server and simply need to install the appropriate Web server plug-in.

Pre-install planning information is covered in 4.5, “Planning for IBM HTTP Server” on page 123.

Note: This section assumes that you are installing the IBM HTTP Server on a machine separate from the WebSphere Application Server. If you are installing both the IBM HTTP Server and WebSphere Application Server on the same machine, skip to 5.3, “Installing IBM WebSphere Application Server” on page 147.

5.2.1 Installation

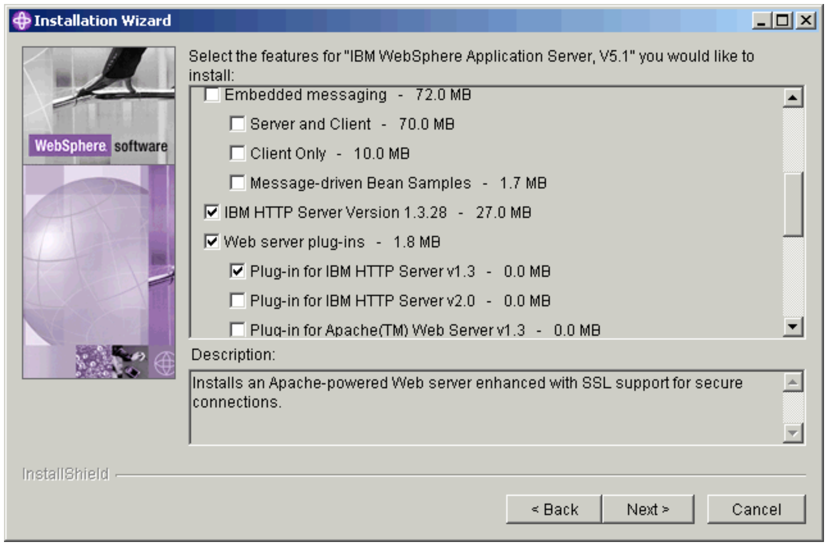
To install, complete the following steps on the Web server machine:

1. Log on as an administrator user in the local server domain (not part of a Windows domain).
2. Insert the IBM WebSphere Application Server 5.1 CD into the CD-ROM drive. This CD-ROM also contains the IBM HTTP Server.
3. Using the Windows Explorer, switch to the \win directory on the CD. Double-click **LaunchPad.bat** to start the install.
4. Select a language for the Launchpad and click **OK**.
5. Review the *Readme* and *Installation Guide*.
6. Click **Install the product**.
7. Select a language to be used for the installation and click **OK**.
8. On the Welcome window, click **Next** to continue.
9. Accept the agreement terms and click **Next**.
10. The installation will verify that your system has the required prerequisites.

    Note: If you are missing prereqs, you will be given a list of what is missing. You should stop at this point and bring the system up to the required level. However, if you choose to continue, the installation wizard will allow you to.

    Once the wizard confirms that all prerequisites have been met, or you elect to ignore the missing prereqs, the wizard will continue.
11. Select **Custom** for the installation type, then select **Next** to continue.
12. A window will display a list of features to install.

    Features to install: The IBM HTTP Server, Web server plug-in, and WebSphere Application Server are all located on the same install media. If you are installing the IBM HTTP Server, select the options shown in Figure 5-1. If you already have a supported Web server installed, or plan to install the IBM HTTP Server on a separate server, just select the appropriate Web server plug-in.



Select the features for "IBM WebSphere Application Server, V5.1" you would like to install:

- [ ] Embedded messaging - 72.0 MB
  - [ ] Server and Client - 70.0 MB
  - [ ] Client Only - 10.0 MB
  - [ ] Message-driven Bean Samples - 1.7 MB
- [✓] IBM HTTP Server Version 1.3.28 - 27.0 MB
- [✓] Web server plug-ins - 1.8 MB
  - [✓] Plug-in for IBM HTTP Server v1.3 - 0.0 MB
  - [ ] Plug-in for IBM HTTP Server v2.0 - 0.0 MB
  - [ ] Plug-in for Apache(TM) Web Server v1.3 - 0.0 MB

Description:

Installs an Apache-powered Web server enhanced with SSL support for secure connections.

Figure 5-1 Installing IBM HTTP Server

Select **Next** to continue.

13. Select the directories to be used for the IBM HTTP Server and Web server plug-in installation. The WebSphere installation directory is used to hold the Web server plug-in and the installation log files. Select **Next** to continue.
14. If you elect to install the server as a Windows service, you will be prompted to provide a user ID and password for starting and stopping the HTTP Server. The user ID value will default to the ID you logged in with. Click **Next** to continue.
15. The Summary window will display the options you have chosen. Select **Next** to begin copying the files.
16. The last step of the installation is registering the product. When done, click **Finish**.

5.2.2 Configuring IBM HTTP Server

After installation of the IBM HTTP Server, the following configuration tasks must be completed:

1. Create IBM HTTP Server admin account.
2. Update httpd.conf.

Creating an administrator user ID and password

In order to use the IBM HTTP Server administration, you will need to identify a user ID and password for the administrator. This user ID and password are stored in a file specific to the IBM HTTP Server. The user ID does not have to be a valid user on the operating system.

From a command window, enter the following:

```
cd <IHS_HOME>
htpasswd -m -c conf\admin.passwd <admin_user_id>
```

When prompted, enter a password for the user and verify it.

> **Note:** The `-c` option says to create a new file. You only use this the first time you enter an admin ID. You can add as many user IDs as you want, but do not specify `-c` after the first time.

For example:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\>cd ibmhttpserver
C:\IBMHttpServer>htpasswd -m -c conf\admin.passwd admin
New password: *****
Re-type new password: *****
Adding password for user admin
C:\IBMHttpServer>htpasswd -m conf\admin.passwd wasadmin
```

Figure 5-2 Adding an IBM HTTP Server administrator user ID

You can add multiple users and passwords.

Updating httpd.conf

The IBM HTTP Server configuration file httpd.conf must have the correct server name. This will most likely be done correctly during the install but check to make sure:

1. Edit the <IHS_HOME>\conf\httpd.conf configuration file using a text editor.
2. Find the ServerName line (probably the first line) and make sure it has the fully qualified DNS name of the server (for example, <hostname.domain.com>).
3. Save the changes and exit.

5.2.3 Verifying the IBM HTTP Server installation

In order to verify the IBM HTTP Server installation, perform the following checks:

1. Check the installation log.
2. Check that services are running.
3. Check request handling.
4. Log in to the administration GUI.

Checking the installation log

Installation messages for the IBM HTTP Server are stored in <WAS_HOME>\logs\ihs_log.txt.

Checking that services are running

:If you installed IBM HTTP Server as a Windows service, verify that the Windows services listed in Table 5-2 have been added and are running.

Table 5-2 IBM HTTP Server Windows services

Service name	Status	Startup mode
IBM HTTP Administration 1.3.28	Started	Automatic
IBM HTTP Server 1.3.28	Started	Automatic

If the services are not running, you can start them by selecting the following:

- **Start -> Programs -> IBM HTTP Server 1.3.28 -> Start HTTP Server**
- **Start -> Programs -> IBM HTTP Server 1.3.28 -> Start Administration Server**

Note: If you didn't install IBM HTTP Server as a service, you can start the server by entering the following:

```
cd <IHS_HOME>
apache
```

The process can be stopped by pressing **Ctrl-C**.

If you want to start the server using a configuration file other than the default at <IHS_HOME>\conf\httpd.conf, enter the following:

```
cd <IHS_HOME>
apache -f "confg\file_name"
```

Note that the file location will be relative to the <IHS_HOME> directory.

Checking request handling

To check IBM HTTP Server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the IBM HTTP Server home page:

   `http://<hostname.domain.com>/`

The window shown in Figure 5-3 will be displayed if the IBM HTTP Server has been installed and configured correctly.

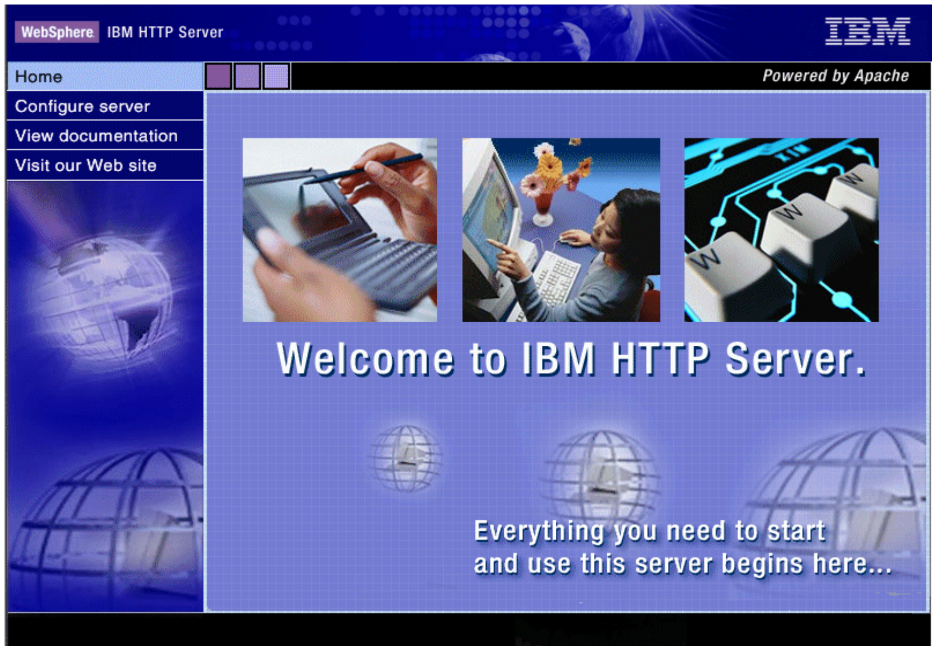


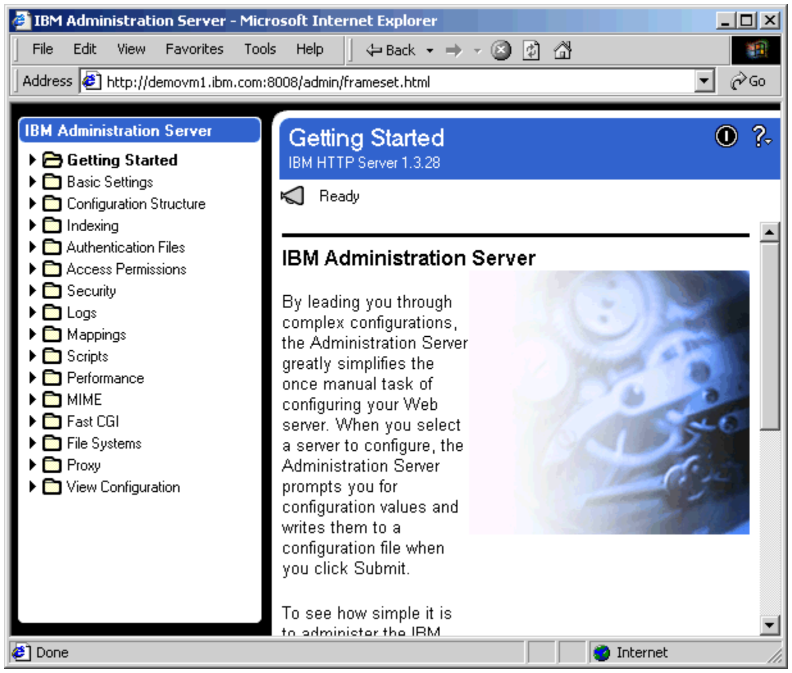
Figure 5-3 IBM HTTP Server Welcome window

Logging in to the administration interface

You can reach the IBM HTTP Server administration server in one of the following ways:

- Click **Configure server** from the Welcome window, as shown in Figure 5-3.
- `http://<hostname>:8008`
- `http://<hostname>/apadminred.html`

You will need to log in using the user ID and password you specified in "Creating an administrator user ID and password" on page 143.



IBM Administration Server

By leading you through complex configurations, the Administration Server greatly simplifies the once manual task of configuring your Web server. When you select a server to configure, the Administration Server prompts you for configuration values and writes them to a configuration file when you click Submit.

To see how simple it is to administer the IBM

Figure 5-4 IBM HTTP Server administration

The configuration of the IBM HTTP Server, except the use of the Web server plug-in, is beyond the scope of this book. Please refer to the documentation available from the Welcome window (see Figure 5-3 on page 145).

5.2.4 Verifyingthe Web server plug-in installation

To verify that the Web server plug-in was successfully installed, open the httpd.conf configuration file. At the bottom of the file you should see the following entry:

```
LoadModule ibm_app_server_http_module
"C:\WebSphere\AppServer/bin/mod_ibm_app_server_http.dll"
WebSpherePluginConfig "C:\WebSphere\AppServer/config/cells/plugin-cfg.xml"
```

You will not be able to verify that the plug-in is functioning correctly until you have installed WebSphere Application Server and generated the Web server plug-in configuration.

Note the designated location of the plug-in configuration specified in httpd.conf, in this case, C:\WebSphere\AppServer/config/cells/plugin-cfg.xml. This is where the plug-in will expect to find its configuration file. When you generate the Web server plug-in configuration from WebSphere Application Server, you will need to copy the generated file to this location.

Generating the Web server plug-in: Before you can use the Web server plug-in with a WebSphere Application Server installation, you will need to generate the plug-in configuration file and move it to the proper location on the Web server. Since this obviously can't be done until WebSphere Application Server has been installed, this topic is covered later in 5.3.6, "Generating the Web server plug-in installation" on page 153.

5.3 Installing IBM WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server 5.1 for Windows 2000.

5.3.1 Installation

To install WebSphere Application Server using the GUI installer interface, complete the following steps:

Note: If you are installing WebSphere Application Server and the Web server plug-in on the Web server machine, stop the Web server during the install. The Web server configuration file will be updated as a part of the installation.

1. Log on as an administrator user in the local server domain (not part of the Windows domain).
2. Insert the IBM WebSphere Application Server 5.1 CD.
3. Start the IBM WebSphere Application Server installation by clicking **LaunchPad.bat** on the root of the CD.
4. Select a language for the Launchpad and click **OK**.
5. Review the *Readme* and *Installation Guide*.
6. Click **Install the product**.
7. Select a language to be used for the installation and click **OK**.
8. On the Welcome window, click **Next** to continue.
9. The software license agreement window will appear. Read the agreement, and accept the terms. Click **Next** to continue.
10. If you have previously installed WebSphere components (for example, the IBM HTTP Server, Web server plug-in, or Deployment Manager), you will be

given the opportunity to add additional features to the current install or to install a new copy. You will also be given the opportunity to select alternate ports.

11. The installation will verify that your system has the required prerequisites.

    Note: If you are missing prereqs, you will be given a list of what is missing. You should stop at this point and bring the system up to the required level. However, if you choose to continue, the installation wizard will allow you to.

    Once the wizard confirms that all prerequisites have been met, or you elect to ignore the missing prereqs, the wizard will continue.

12. Select the **Custom** radio button and **Next** to continue.
13. Select the features to install.
14. The next window will enable the user to select the installation directories for IBM WebSphere Application Server, Embedded Messaging server and client, and IBM HTTP Server. Verify the install directories are correct and select **Next** to continue.
15. The next window asks for the node name for this installation. The defaults are probably fine. However, keep in mind that the node name must be unique within a cell. The node name is a logical name, so although the default is the system host name, it doesn't actually have to be the host name. Click **Next** to continue.
16. The next window will allow you to elect to run the IBM HTTP Server and/or WebSphere Application Server as services. Windows services can be automatically started, and startup and recovery operations can be configured.

    If you elect to have either run as a service, you will need to enter a user ID and password for the service(s) to run under. The user ID must be valid. The default is the user ID you are logged on with. If the ID doesn’t have the proper authority, the installation will assign user rights that allow it to work properly.

    Note: The user ID should be less than 12 characters to satisfy a WebSphere MQ limitation.

17. The last window before installation begins shows a summary of all of the choices that were made in the custom installation. Select **Next** to begin the installation. A progress bar will indicate the status of installation.
18. When all of the files have been copied, the installer will begin the process of installing the sample applications that were selected during installation. It will install and start each application during this time and show a status bar indicating the progress.

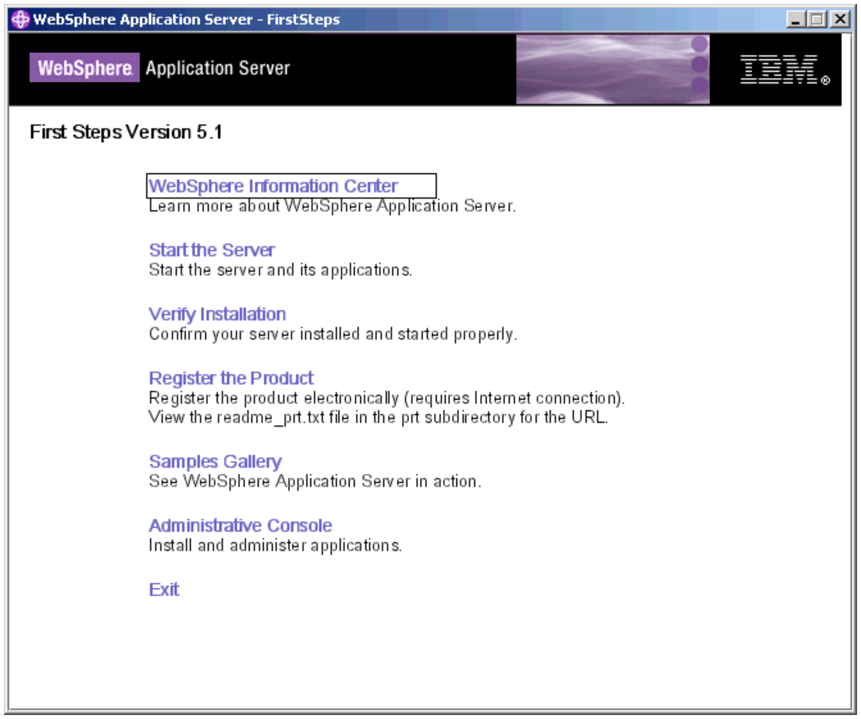
19. Once the installation is complete the installer will prompt the user to register. The user can choose to register the product immediately by checking the check box at the bottom of the window, or manually register at a later time by deselecting the check box.

20. Click **Next** to finish.

If you have any errors, check the <WAS_HOME>\logs\log.txt for installation messages. There are separate install logs in the same directory for each component.

5.3.2 Verifying the IBM WebSphere Application Server installation

The installation will complete by starting the First Steps window. From there, you can start the server and run the installation verification.


WebSphere Application Server - FirstSteps
WebSphere Application Server
IBM

First Steps Version 5.1

WebSphere Information Center
Learn more about WebSphere Application Server.

Start the Server
Start the server and its applications.

Verify Installation
Confirm your server installed and started properly.

Register the Product
Register the product electronically (requires Internet connection).
View the readme_prt.txt file in the prt subdirectory for the URL.

Samples Gallery
See WebSphere Application Server in action.

Administrative Console
Install and administer applications.

Exit

Figure 5-5 First Steps window

From the First Steps window:

1. Select **Start the server**. The server initialization messages will appear in the message box on the First Steps window. When the server has successfully started you will see the following message:

   ```
   ADMU3000I: Server server1 open for e-business; process id is 2928
   ```

   If the server fails to initialize, check the startServer.log and SystemOut.log in the <WAS_HOME>/logs/server1 directory.

2. Select **Verify Installation** in the First Steps window to run the Installation Verification Test (IVT) program. The verification messages will appear in the message box. If the verification is successful, you will see the following messages:

   ```
   IVTL0050I: Servlet Engine Verification Status - Passed
   IVTL0055I: JSP Verification Status - Passed
   IVTL0060I: EJB Verification Status - Passed
   IVTL0070I: IVT Verification Succeeded
   IVTL0080I: Installation Verification is complete
   ```

   If you have problems, check the ivt.log file in <WAS_HOME>\logs.

Note: After installation, the First Steps window can be opened at any time in one of the following ways:

- Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> First Steps**
- Run **`<WAS_HOME>\firststeps\firststeps.bat`**

If you have any problems starting the server or during the verification, check the installation messages stored in the log files. The following installation log files can be found in <WAS_HOME>\logs.

Table 5-3 Installation log files

Component	**File**
WebSphere Application Server	log.txt
IBM HTTP Server	ihs_log.txt
Default Application	installDefaultApplication.log
Sample Application	installSamples.log
Administrative console	installAdminConsole.log
MDB Samples Application	installMessagingSamples.log
Pet Store Application	installPetStore.log

Component	File
Plants By WebSphere Application	installPlantsByWeb.log
Technology Samples Application	installTechSamples.log
Web Services Samples Application	installWebServices.log
IVT Application	installIVTApp.log

5.3.3 Starting and stopping server1

WebSphere Application Server is installed with one server (“server1”) preconfigured. The application that provides the administrative console is installed on this server, as well as the sample applications. You can install additional applications on this server and you can create new servers.

Before accessing the administrative console or applications that are installed on the server, you will need to start server1. You have seen how to do this from the First Steps window, but after installation, it is unlikely that you will use the First Steps. For future reference, you can start or stop server1 in one of the following ways:

- Using the Windows Start menu:
  - Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Start the Server**
  - Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Stop the Server**
- From a command prompt, run:
  - `<WAS_HOME>\bin\startServer server1`
  - `<WAS_HOME>\bin\stopServer server1`
- Using the Windows services window. Start or stop the IBM WebSphere Application Server V5 - server1 service. By default the service is set to be started manually, but you can change this to `Automatic`.

Network Deployment: If the server is a part of a Network Deployment cell and the node agent is active, you can also use the administrative console to start and stop servers.

Check the log files, SystemOut.log and SystemErr.log, under the <WAS_HOME>/logs/server1 directory. The following message should appear in the SystemOut.log file indicating a successful start.

```
WSVR0001I: Server server1 open for e-business
```

5.3.4 Accessing the administrative console

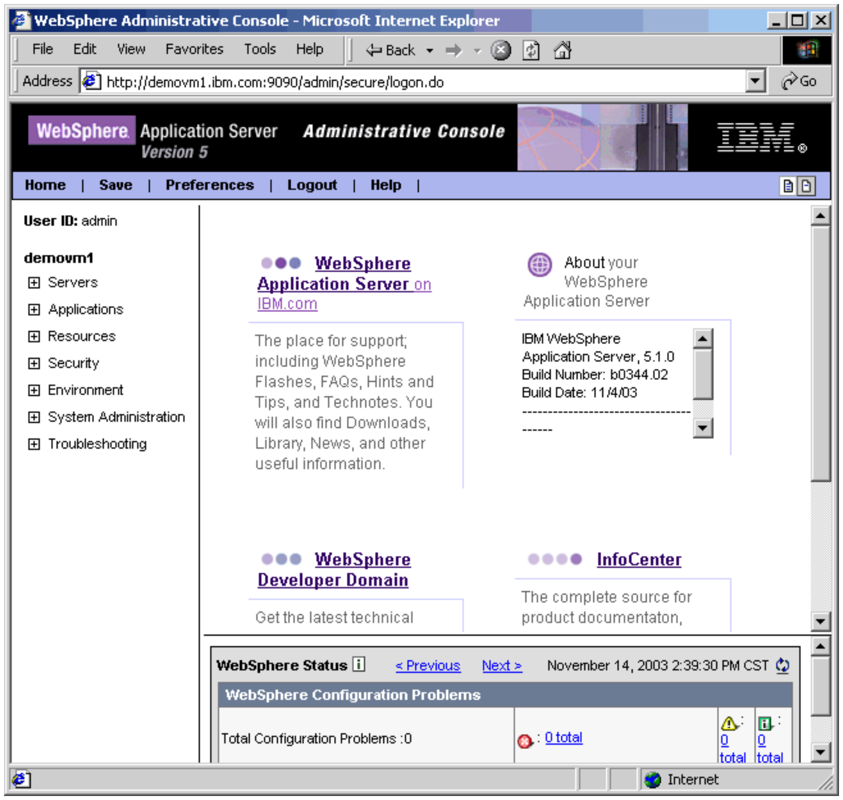
You can access the administrative console in any one of the following ways:

- Using a Web browser, request the following URL:

  `http://<hostname>:9090/admin`

- Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Administrative Console** from the First Steps menu.

You will be asked to enter a user ID to log in. Until you enable console security, this user ID is for tracking purpose only, so any character string is acceptable. The initial administrative console window can be seen in Figure 5-6.



WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://demovm1.ibm.com:9090/admin/secure/logon.do
Go
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
demovm1
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
About your WebSphere Application Server
WebSphere Developer Domain
InfoCenter
WebSphere Status
< Previous
Next >
WebSphere Configuration Problems
0 total
Internet

The place for support, including WebSphere Flashes, FAQs, Hints and Tips, and Technotes. You will also find Downloads, Library, News, and other useful information.

IBM WebSphere Application Server, 5.1.0
Build Number: b0344.02
Build Date: 11/4/03

Get the latest technical

The complete source for product documentaton,

November 14, 2003 2:39:30 PM CST

Total Configuration Problems :0

Figure 5-6 Administrative console

5.3.5 Accessing applications on WebSphere

When you install WebSphere Application Server you automatically have at least one application installed. The application, called DefaultApplication, exists simply to help you verify that the installation is working. You can access the servlets in DefaultApplication with the following URLs:

- `http://<WAS_hostname>:9080/snoop`
- `http://<WAS_hostname>:9080/hello`
- `http://<WAS_hostname>:9080/hitcount`

If you installed the sample applications, you can access them with the following URL:

```
http://<WAS_hostname>:9080/WSsamples
```

Note that in each case, the URL specifically designates port 9080. This is the default HTTP transport port. If you changed this during installation you will need to substitute the correct port number. Once you have set up a Web server and the Web server plug-in, you will be able to access these applications through the Web server and will not need to designate a port.

5.3.6 Generating the Web server plug-in installation

Network Deployment: If you are going to install Network Deployment and will be adding this WebSphere Application Server to a cell, you can skip this step and do it after the cell is set up.

To generate and verify the Web server plug-in, you will need to do the following:

- Verify that the plug-in statements have been added to the Web server configuration file.

3. Generate the configuration file from WebSphere Application Server. Using the administrative console, expand **Environment** and select **Update Web Server Plugin**. Click **OK** to generate the configuration file.

- Copy the configuration to the Web server system. The location you need to copy it to is specified in the WebSpherePluginConfig directive of the Web server configuration file.
- Test the connection to WebSphere through the Web server. To test the remote Web server connection to WebSphere, access the snoop servlet using the following URL:

  ```
  http://<Web server hostname>/servlet/snoop
  ```

5.3.7 Implementing security

Network Deployment: If you are going to install Network Deployment and will be adding this WebSphere Application Server to a cell, you can skip this step and do it after the cell is set up.

Security is an important part of any e-business environment. Designing and implementing the security structure is a topic of its own and is addressed in detail in *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

However, at this time, you should consider securing the administrative console to restrict who has access to the WebSphere configuration. The instructions to do this are in 8.5, “Securing the administrative console” on page 276.

5.4 Installing IBM WebSphere Application Server Network Deployment

The following procedure uses our existing base installation to upgrade to a Network Deployment installation. It first addresses the installation steps, then outlines the migration steps necessary to preserve any base configuration changes made prior to the Network Deployment installation.

The installation procedure for IBM WebSphere Application Server Network Deployment is as follows:

Note: If you are installing Network Deployment on the same system as IBM WebSphere Application Server V5, you will get port conflicts for the administration ports. If you will be adding the WebSphere Application Server to the Network Deployment cell, you do not need to be concerned with this. The WebSphere Application Server administrative console is an enterprise application. It is deleted from the configuration when you add the node to the cell and the Network Deployment administrative console application becomes the configuration interface for all the nodes in the cell. Until then, only one administrative console application can be active at a time.

1. Log on as an administrator user in the local server domain (not part of the windows domain).
2. Insert the Network Deployment CD.
3. Start the installation by clicking **LaunchPad.bat** in the \win directory.
4. Select a language for the LaunchPad and click **OK**.

5. Review the *Readme* and *Installation Guide*.
6. Click **Install the product**.
7. Select the language to be used for the installation, select **OK**.
8. Select **Next** on the Welcome window to continue.
9. The License Agreement will be displayed. Read the License Agreement and if you agree to its terms, accept them, and select **Next** to continue.
10. If you have previously installed WebSphere components (for example, the IBM HTTP Server, Web server plug-in, or WebSphere Application Server), you will be given the opportunity to add additional features to the current install or to install a new copy. You will also be given the opportunity to select alternate ports.

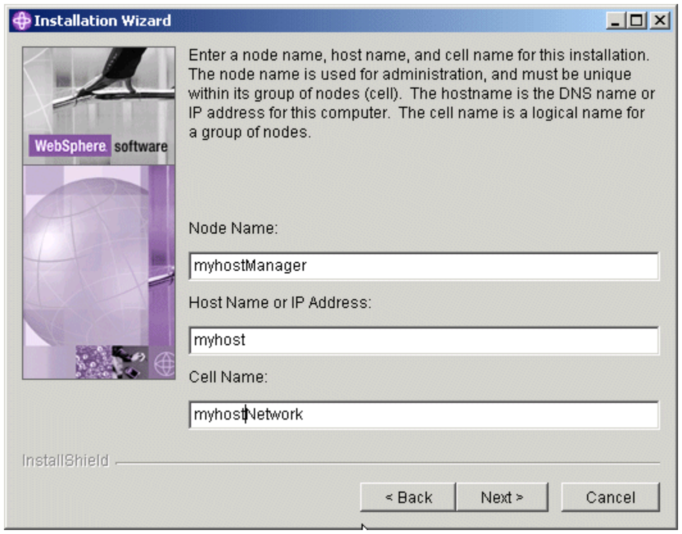
    After resolving the ports, select **Next** to continue.
11. The installation will verify that your system has the required prerequisites.

    Note: If you are missing prereqs, you will be given a list of what is missing. You should stop at this point and bring the system up to the required level. However, if you choose to continue, the installation wizard will allow you to.

    Once the wizard confirms that all prerequisites have been met, or you elect to ignore the missing prereqs, the wizard will continue.
12. The next window will display the features that are available for install. Choose the features to install and click **Next** to continue.
13. The next window will prompt for a directory to install the Deployment Manager to. Take the default or change it to meet your installation standards. Click **Next.**
14. The next window will prompt for a WebSphere Application Server node name, network host name, and cell name for your installation.

    The node name must be unique in the cell. If you select a name that is not unique, the install will continue, but you will have problems later when you attempt to add the node to the cell. The defaults use the host name of your machine as a basis.

    Select **Next** to continue.



Enter a node name, host name, and cell name for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer. The cell name is a logical name for a group of nodes.

Node Name: myhostManager

Host Name or IP Address: myhost

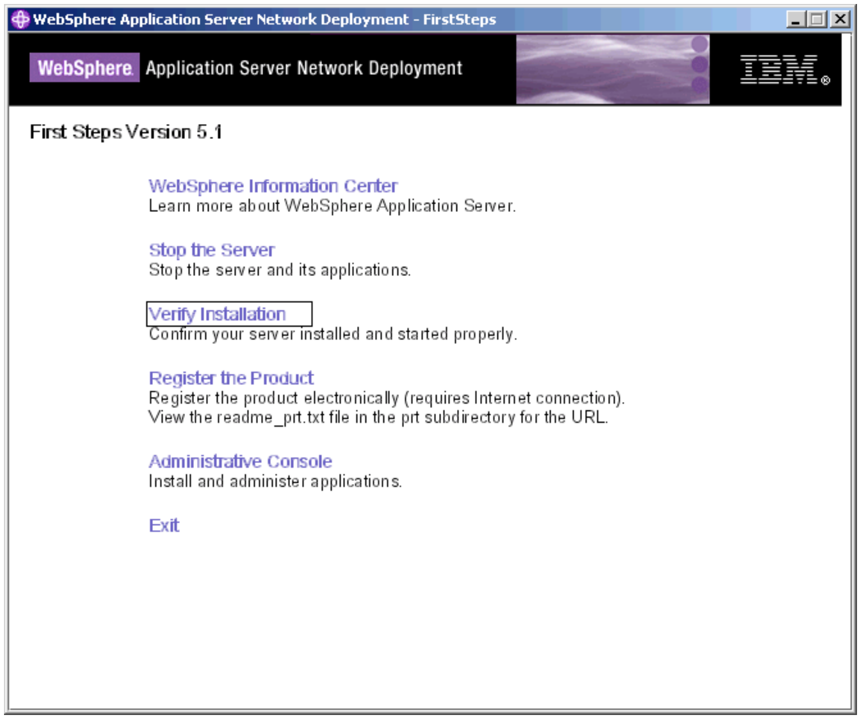
Cell Name: myhostNetwork

Figure 5-7 Node and host name selection

15. In the next window, you can choose to run the Deployment Manager as a Windows service. If you do choose this option, you will need to enter a user ID and password for the Deployment Manager services. Make the appropriate selection and click **Next** to continue.
16. The last window before installation is the Summary window. The window will detail the sections that were made during the install. Verify that these selections are correct, and select **Next** to begin copying files to the machine.
17. You will be given a chance to register the product. After doing so, click **Finish** to exit the install.

5.4.1 Verifying the Deployment Manager installation

As the last step of the installation, the First Steps window is opened. From this window you can view the InfoCenter, start and stop the Deployment Manager, run the installation verification tests, and access the administrative console.


WebSphere Application Server Network Deployment - FirstSteps

WebSphere Application Server Network Deployment

First Steps Version 5.1

WebSphere Information Center
Learn more about WebSphere Application Server.

Stop the Server
Stop the server and its applications.

Verify Installation
Confirm your server installed and started properly.

Register the Product
Register the product electronically (requires Internet connection).
View the readme_prt.txt file in the prt subdirectory for the URL.

Administrative Console
Install and administer applications.

Exit

Figure 5-8 Network Deployment First Steps window

After installation the Deployment Manager is automatically started.

Select **Verify Installation** in the First Steps window. The Installation Verification Test (IVT) program will execute. The verification messages will appear in the message box. If the verification is successful, you will see the following messages:

```
IVTL0095I: defaulting to host carlasr31 and port 9090
IVTL0010I: Connecting to the WebSphere Application Server carlasr31 on
port: 9090
IVTL0015I: WebSphere Application Server carlasr31 is running on port: 9090
IVTL0070I: IVT Verification Succeeded
IVTL0080I: Installation Verification is complete
```

Note: After installation, the First Steps window can be opened at any time in one of the following ways:

- Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Network Deployment -> First Steps**
- Run **`<WAS_ND_HOME>\firststeps\firststeps.bat`**

If you have any problems starting the server or during the verification, check the installation messages stored in the log files. The following installation log files can be found in <WAS_ND_HOME>\logs.

5.4.2 Starting and stopping the Deployment Manager

The Deployment Manager is started automatically after installation and as you have seen can be started and stopped from the First Steps window. However, it is unlikely that you will use the First Steps after the installation. For future reference, you can start or stop the Deployment Manager in one of the following ways:

- Using the Windows Start menu:
  - Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Network Deployment -> Start the Deployment Manager**
  - Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Network Deployment -> Stop the Deployment Manager**
- From a command prompt, enter:
  - `<WAS_ND_HOME>\bin\startManager`
  - `<WAS_ND_HOME>\bin\stopManager`
- Using the Windows services window. Start or stop the IBM WebSphere Application Server V5 - dmgr service. By default the service is set to be started manually, but you can change this to Automatic.

5.4.3 Accessing the administrative console

The administrative console can be accessed in one of the following ways:

- Using URL: `http://<ND_hostname>:9090/admin`
- Select **Start -> Programs -> IBM WebSphere -> Application Server V5.0 -> Network Deployment -> Administrative Console**
- From the First Steps window.

You will be asked to enter a user ID to log in. Until you enable console security, this user ID is for tracking purpose only, so any character string is acceptable.

5.4.4 Adding nodes to the cell

You have just been through the process of installing Network Deployment. This gives you a cell framework, but as of yet, there are no nodes in the cell. Presumably you have also installed one or more WebSphere Application Servers. If not, you should do this now.

In order for a node to be managed by the Deployment Manager, it must be added to the cell.

1. On the Deployment Manager node, start the Deployment Manager.

   ```
   cd <WAS_ND_HOME>\bin
   startManager
   ```

2. On the WebSphere Application Server node, stop all the servers. To do this open a command prompt and check the status of the servers with the following commands:

   ```
   cd <WAS_HOME>\bin
   serverStatus -all
   ```

   If this is a new install you should only have one server, server1. If server1 is started, stop it with the following command:

   ```
   stopServer server1
   ```

3. On the WebSphere Application Server node, add the node to the cell with the following commands:

   ```
   addNode <DM_hostname> 8879 -includeapps
   ```

   Note that the **`addNode`** command is run from the application server side, not the Network Deployment side. The `<DM_hostname>` is the host name of the Network Deployment machine and the port (`8879`) is the SOAP connector port for the Deployment Manager.

   Finding the SOAP connector port: If you need to look up the SOAP connector port for the Deployment Manager, do the following:

   1. Open the administrative console for the Deployment Manager.
   2. Select **System Administration -> Deployment Manager**.
   3. Select **End Points** in the Additional Properties table.
   4. Select **SOAP_CONNECTOR_ADDRESS.**

   When the process is done you should get the following message:

   ```
   Node nodename has been successfully federated.
   ```

Example 5-1 shows an example of this process.

Example 5-1 Adding a node to a cell

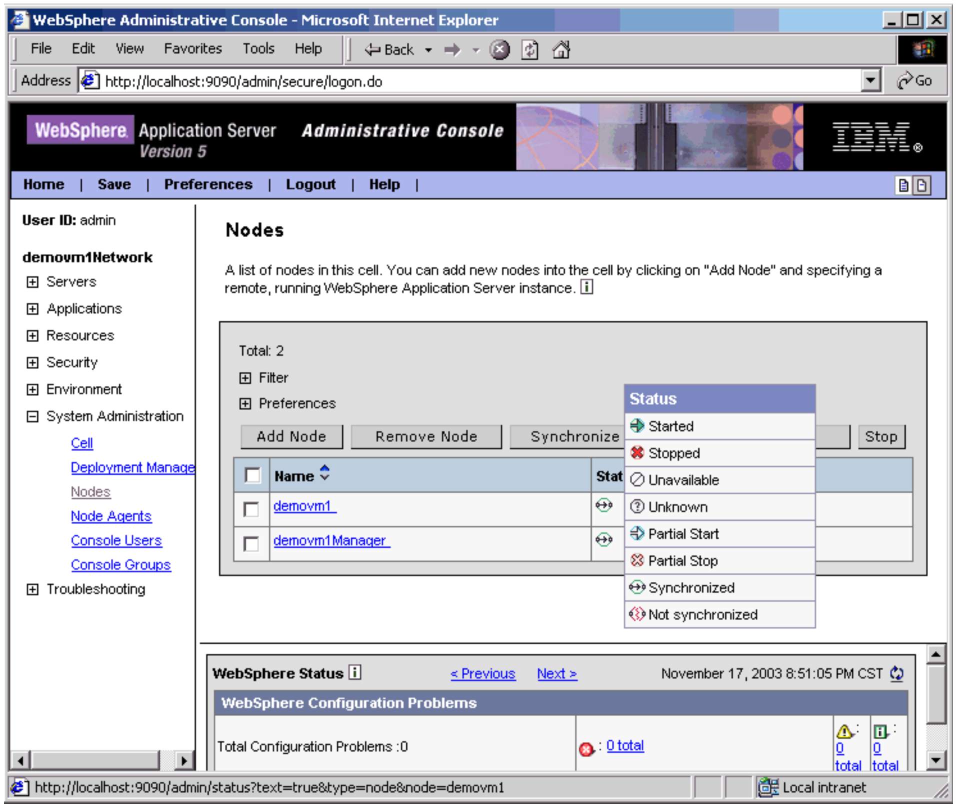
```
C:\WebSphere\AppServer\bin>serverStatus -all
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\serverStatus.log
ADMU0500I: Retrieving server status for all servers
ADMU0505I: Servers found in configuration:
```

```
ADMU0506I: Server name: server1
ADMU0508I: The Application Server "server1" is STARTED

C:\WebSphere\AppServer\bin>stopserver server1
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\server1\stopServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

C:\WebSphere\AppServer\bin>addnode localhost 8879 -includeapps
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\addNode.log
ADMU0001I: Begin federation of node carlasr31 with Deployment Manager at
           localhost:8879.
ADMU0009I: Successfully connected to Deployment Manager Server: localhost:8879
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU2010I: Stopping all server processes for node carlasr31
ADMU0512I: Server server1 cannot be reached. It appears to be stopped.
ADMU0024I: Deleting the old backup directory.
ADMU0015I: Backing up the original cell repository.
ADMU0012I: Creating Node Agent configuration for node: carlasr31
ADMU0014I: Adding node carlasr31 configuration to cell: carlasr31Network
ADMU0016I: Synchronizing configuration between node and cell.
ADMU0018I: Launching Node Agent process for node: carlasr31
ADMU0020I: Reading configuration for Node Agent process: nodeagent
ADMU0022I: Node Agent launched. Waiting for initialization status.
ADMU0030I: Node Agent initialization completed successfully. Process id is:
           2556
ADMU0523I: Creating Queue Manager for node carlasr31 on server jmsserver
ADMU0525I: Details of Queue Manager creation may be seen in the file:
           createMQ.carlasr31_jmsserver.log
ADMU9990I:
ADMU0300I: Congratulations! Your node carlasr31 has been successfully
           incorporated into the carlasr31Network cell.
ADMU9990I:
ADMU0306I: Be aware:
ADMU0302I: Any cell-level documents from the standalone carlasr31 configuration
           have not been migrated to the new cell.
ADMU0307I: You might want to:
ADMU0303I: Update the configuration on the carlasr31Network Deployment Manager
           with values from the old cell-level documents.
ADMU9990I:
ADMU0003I: Node carlasr31 has been successfully federated.
```

You can see the new node in the Network Deployment administrative console by expanding System Administration and clicking **Nodes**. Check the status of the new node and make sure it is synchronized. If not, select the new node and click **Synchronize**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
demovm1Network
Servers
Applications
Resources
Security
Environment
System Administration
Cell
Deployment Manage
Nodes
Node Agents
Console Users
Console Groups
Troubleshooting
Nodes
Total: 2
Filter
Preferences
Add Node
Remove Node
Synchronize
Stop
Name
Stat
demovm1
demovm1Manager
Status
Started
Stopped
Unavailable
Unknown
Partial Start
Partial Stop
Synchronized
Not synchronized
WebSphere Status
< Previous
Next >
November 17, 2003 8:51:05 PM CST
WebSphere Configuration Problems
Total Configuration Problems :0
0 total
http://localhost:9090/admin/status?text=true&type=node&node=demovm1
Local intranet

A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote, running WebSphere Application Server instance.

Figure 5-9 Display the nodes in the cell

Starting the node agent: After adding a node to a cell, a node agent server is required to be running on the node in order for the Deployment Manager to communicate with it. The node agent is started automatically during the addnode process but subsequently must be started manually by entering the following from a command prompt on the WebSphere Application Server machine:

```
cd <WAS_HOME>\bin
startnode
```

Starting server1: Once the node agent is started, you can start the server from the Deployment Manager administrative console. Expand **Servers** and select **Application Servers** to see server1. Select (check the box) **server1** and click **Start**.

5.4.5 Accessing applications

Installing a Network Deployment environment and adding a node to a cell does not change the basic capabilities of the server. It only changes the way the server is managed.

5.5 Installing WebSphere Application Server - silent mode

This section describes how to install WebSphere Application Server using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

Planning and post-installation activities for silent installs remain the same as with a GUI installation and will not be addressed here.

Note: IBM HTTP Server is supplied with WebSphere Application Server. If you plan to use a different Web server, you must purchase it and install it separately. It is recommended that the Web server be installed before WebSphere Application Server.

5.5.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server as a template for creating a customized response file.

With default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.4.1
- IBM HTTP Server 1.3.28
- IBM WebSphere Application Server 5.1
- Embedded messaging
- Performance and analysis tools
- WebSphere Application Server application samples
- Documentation in U.S. English

Note: All products except IBM HTTP Server are installed in the directory WebSphere\AppServer. IBM HTTP Server is installed in the directory \IBMHttpServer. In addition, WebSphere Application Server is configured for use with IBM HTTP Server when you use the default response file.

5.5.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components of WebSphere Application Server or to install the products in a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

The following lines are an example of which lines you might want to update:

```
-P wasBean.installLocation="C:\Program Files\WebSphere\AppServer"
-P ihsFeatureBean.installLocation="C:\Program Files\IBMHTTPServer"
-P mqSeriesServerBean.installLocation="C:\Program Files\IBM\WebSphere MQ"
-P mqSeriesClientBean.installLocation="C:\Program Files\IBM\WebSphere MQ"
-W nodeNameBean.nodeName="mynode1"
-W nodeNameBean.hostName="9.37.37.2"
-W defaultIHSConfigFileLocationBean.value="C:\IBMHTTPServer\conf\httpd.conf"
```

5.5.3 Performing a silent installation

Perform the following steps to create a customized response file to install WebSphere Application Server. These instructions assume that the installation is being performed from the product CD-ROM:

1. Ensure that you are logged into the machine with administrator privileges.
2. If a Web server is running on your system, stop the Web server. If you plan to install IBM HTTP Server 1.3.28 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.28 on your system, you must uninstall it for the WebSphere Application Server installation program to install IBM HTTP Server 1.3.28.
3. Load the IBM WebSphere Application Server V5.1 CD-ROM into the CD-ROM drive and navigate to the \win directory.
4. Copy responsefile.txt to your local file system.
5. Customize the new file, using the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation. See “Customized response file” on page 163 for minimum changes of the response file.

6. Run the installation script from the install media by using the following command:

   ```
   install -options <new_responsefile.txt>
   ```

   The install script uses the response file to install the components and options that you have selected.

7. After installation is complete, review the log.txt file located in the <WAS_HOME>/logs directory to determine if the silent installation was successful.

5.6 Installing Network Deployment - silent mode

This section describes how to install WebSphere Application Server Network Deployment using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script for WebSphere Application Server Network Deployment, supplying the response file as a command-line parameter.

These instructions assume that your machine has sufficient memory and disk space for your installation. Planning and post-installation activities for silent installs remain the same as with a GUI installation and will not be addressed here.

5.6.1 Default response file

A default response file, named responsefile.txt, is supplied with Network Deployment. With default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.4.1
- Deployment Manager
- Embedded messaging
- Web services

5.6.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components of Network Deployment or to install the products in a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

The following lines are an example of which lines should be updated at least for the basic installation:

```
-P wasBean.installLocation="C:\WebSphere\DeploymentManager"
-W nodeNameBean.nodeName="m10df51fManager"
-W nodeNameBean.cellName="m10df51fNetwork"
-W nodeNameBean.hostName="m10df51f"
```

In the example, the following items are specified in the response file respectively.

- Network Deployment installation directory
- Node name
- Cell name
- Host name or IP address

Important: All values should be enclosed in double quotes (““).

5.6.3 Performing a silent installation

Perform the following steps to create a customized response file (if desired) to install Network Deployment. These instructions assume that the installation is being performed from the product CD-ROM:

1. Log on with administrator privileges.
2. Load the IBM WebSphere Application Server Network Deployment 5.1 CD-ROM into the CD-ROM drive.
3. Navigate to the \win directory and create a copy of the default response file, responsefile.txt, on your local file system.
4. Customize the new response file, using the detailed comments throughout the file to help you select the appropriate options for your installation.
5. Run the installation script from the installation media by using the following command:

   ```
   install -options <new_responsefile.txt>
   ```

   The install script uses the response file to install the components and options that you have selected.
6. After installation is completed, review the log.txt log file to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_ND_HOME>/logs.

6

AIX installation steps

This chapter provides detailed procedures for installing, configuring, and verifying IBM HTTP Server, WebSphere Application Server, and WebSphere Network Deployment on AIX. Before beginning your installation, read Chapter 4, “Installation approach” on page 117, which leads you through the planning process necessary for a successful install. The topics in this chapter are:

- Installation of the IBM HTTP Server and /or the Web server plug-in
- Installation of WebSphere Application Server
- Installation of Network Deployment

The intent of this chapter is to lead you through a simple, first-time installation. It should help you understand the basic process of installing a base or Network Deployment environment, accessing the default and sample applications, and accessing the administration facilities.

If you have any issues with coexistence or migration, please refer to the Installation Guide provided with the product.

6.1 Product installation root variables

The variables listed in Table 6-1 are used frequently throughout this documentation to represent the root installation directories of the software components.

Table 6-1 Product Installation roots

Variable	Default value	Component
<WAS_HOME>	/usr/WebSphere/AppServer	WebSphere Application Server
<WAS_ND_HOME>	/usr/WebSphere/Deployment Manager	WebSphere Application Server Network Deployment
<IHS_HOME>	/usr/IBMHttpServer	IBM HTTP Server
<PLUGIN_HOME>	/usr/WebSphere/AppServer	Web server plug-in

Note: Both WAS_HOME and WAS_ND_HOME are defined as WAS_HOME in setupCmdLine.sh under the bin directory of their installation directories.

6.2 Install AIX

Prior to installing any of the WebSphere components, the proper level of the operating system must be installed. For the most current information on the supported software releases, operating systems, and maintenance levels, see:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

To determine whether the server currently has the required maintenance release (or newer) installed, issue the following command:

```
# oslevel -r
```

The command will generate output that resembles the following:

```
5100-02
```

The last two digits, `02` in this example, represent the maintenance level. If your system does not meet the minimum required maintenance level, bring it up to the required level before proceeding.

Tip: The AIX maintenance releases can be obtained from the following IBM site:

`http://www-912.ibm.com/eserver/support/fixes/fcgui.jsp`

or the following IBM FTP sites:

- `ftp://ftp.software.ibm.com/aix/fixes/51/ml` for AIX 5.1
- `ftp://ftp.software.ibm.com/aix/fixes/52/ml` for AIX 5.2

For an advance look at what the installation wizard will check for, you can look at the /aix/waspc/prereqChecker.xml file on the installation media. This file is used during installation to determine if the system has met the requirements for installation. To determine whether a fileset is installed on your system, use the following command:

```
# lslpp -L | grep <fileset>
```

During our test installation, the prereq checker called out for the X11.fnt.ucs.ttf fileset (used for character display).

If the output does not include the required version of the fileset (or newer), then the fileset must be upgraded before continuing.

6.3 Installing the IBM HTTP Server and/or Web server plug-in

This section provides instructions for installing, configuring, and verifying IBM HTTP Server V1.3.28 for AIX and the Web server plug-in. These instructions also apply if you have an existing Web server and simply need to install the appropriate Web server plug-in.

Pre-install planning information is covered in 4.5, “Planning for IBM HTTP Server” on page 123.

Note: This section assumes that you are installing the IBM HTTP Server on a machine separate from the WebSphere Application Server. If you are installing both the IBM HTTP Server and WebSphere Application Server on the same machine, skip to 6.4, “Installing WebSphere Application Server” on page 176.

6.3.1 Installation

Space requirements: The IBM HTTP Server and Web server plug-in both install in the /usr file system. The IBM HTTP Server will require 27 MB of space. The Web server plug-in will require 2.9 MB.

The IBM HTTP Server, Web server plug-in, and WebSphere Application Server all reside on the same installation media. To install the IBM HTTP Server and/or the Web server plug-in, complete the following steps on the Web server machine:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V5.1 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```
4. Change the directory to /mnt.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set using the following commands:

   ```
   export DISPLAY=<IP address of host>:0.0
   export TERM=vt100
   ```
6. Run the LaunchPad.sh installation script:

   ```
   # ./LaunchPad.sh
   ```
7. Select a language for the LaunchPad and click **OK**.
8. Review the *Readme* and *Installation Guide*.
9. Click **Install the product**.
10. Select a language to be used for the installation and click **OK**.
11. On the Welcome window, click **Next** to continue.
12. The software license agreement window will appear. Read the agreement, and accept the terms. Click **Next** to continue.
13. If you have previously installed WebSphere components on this machine (for example, WebSphere Application Server), you will be given the opportunity to add additional features to the current install or to install a new copy. You will also be given the opportunity to select alternate ports.
14. The installation will verify that your system has the required prerequisites.

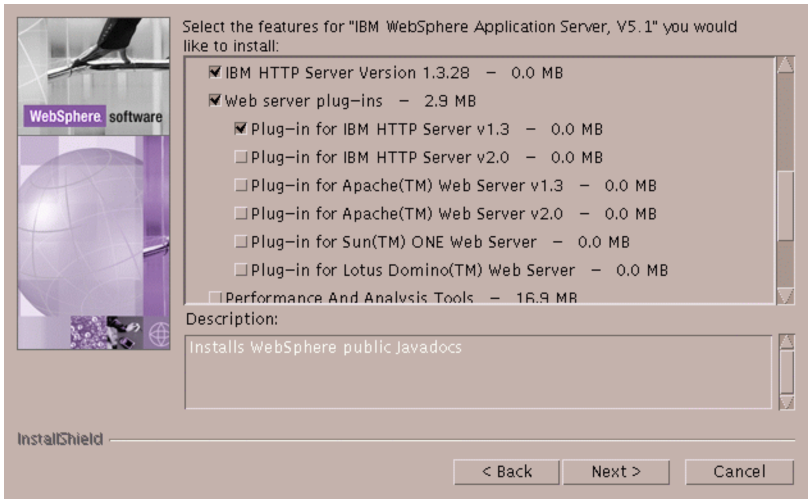
Note: If you are missing prerequisites, you will be given a list of what is missing. You should stop at this point and bring the system up to the required level. However, if you choose to continue, the installation wizard will allow you to.

Once the wizard confirms that all prerequisites have been met, or you elect to ignore the missing prereqs, the wizard will continue.

15. WebSphere provides two installation options, full and custom. Unless this is a test machine, we suggest you use the custom install.

    Select the **Custom** radio button and **Next** to continue.

Features to install: If you are installing the IBM HTTP Server, select the options shown in Figure 6-1. If you already have a supported Web server installed, just select the appropriate Web server plug-in.



Select the features for "IBM WebSphere Application Server, V5.1" you would like to install:

- ☑ IBM HTTP Server Version 1.3.28 – 0.0 MB
- ☑ Web server plug-ins – 2.9 MB
  - ☑ Plug-in for IBM HTTP Server v1.3 – 0.0 MB
  - ☐ Plug-in for IBM HTTP Server v2.0 – 0.0 MB
  - ☐ Plug-in for Apache(TM) Web Server v1.3 – 0.0 MB
  - ☐ Plug-in for Apache(TM) Web Server v2.0 – 0.0 MB
  - ☐ Plug-in for Sun(TM) ONE Web Server – 0.0 MB
  - ☐ Plug-in for Lotus Domino(TM) Web Server – 0.0 MB
- ☐ Performance And Analysis Tools – 16.9 MB

Description:

Installs WebSphere public Javadocs

Figure 6-1 Installing IBM HTTP Server

16. Click **Next**.

17. In the Install directory window, enter the directories for the Web server plug-in (installed in the WebSphere Application Server directory) and the IBM HTTP Server. The defaults are:

    – /usr/WebSphere/AppServer for the Web server plug-in
    – /usr/IBMHttpServer for the IBM HTTP Server

Click **Next** to continue.

18. The last window before installation begins shows a summary of all of the choices that were made in the custom installation. Select **Next** to begin the installation. A progress bar will indicate the status of installation.
19. Once the installation is complete, the installer will prompt you to register the product. You can choose to do this immediately by checking the check box at the bottom of the window, or manually register at a later time by deselecting the check box.
20. Click **Next** to continue and then **Finish**.

6.3.2 Configuring the IBM HTTP Server

After installation, the following configuration tasks must be completed on the IBM HTTP Server machine:

1. Create the IBM HTTP Server admin account.
2. Create the UNIX runtime account.
3. Update httpd.conf.
4. Restart the IBM HTTP Server.

Creating the HTTP server admin account

The administration account is used to access the HTTP Administration Server Configuration GUI. To create the account, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Change the directory to the <IHS_HOME>/bin directory.
4. Create the administration user by typing the following commands:

```
# ./htpasswd -c -m ../conf/admin.passwd admin
New password: <admin_password>
Re-type new password: <admin_password>
```

Where `admin` is the IBM HTTP Server administration user ID.

Note: Both user ID and password for HTTP Administration Server are created here. They are used only in order to log in to the HTTP Administration Server. The user ID created here is not an AIX user ID.

Updating httpd.conf

The IBM HTTP Server configuration file httpd.conf must be updated to reflect the fully qualified host name of the server. To update the IBM HTTP Server configuration file, complete the following steps:

1. Edit the <IHS_HOME>/conf/httpd.conf file and update the settings listed in Table 6-2.

Table 6-2 httpd.conf required settings

Setting	Required value...
ServerName	<hostname.domain.com>

2. Save the changes and exit.

Restarting the IBM HTTP Server

The IBM HTTP Server must be restarted in order for the configuration changes for httpd.conf file to take effect:

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <IHS_HOME>/bin
# ./apachectl stop
# ./apachectl start
```

Restarting the HTTP Administration Server

The HTTP Administration Server must be restarted in order to recognize the configuration changes of httpd.conf:

1. Log in as root.
2. Start a terminal session.
3. Issue the following commands:

```
# cd <IHS_HOME>/bin
# ./adminctl stop
# ./adminctl start
```

6.3.3 Verifying the IBM HTTP Server installation

In order to verify the installation, perform the following checks on the IBM HTTP Server machine:

1. Check the process status.
2. Check request handling.

Checking the process status

To check IBM HTTP Server process status, perform the following steps:

1. Check that the HTTP Server processes are running by issuing the following command:

   ```
   # ps -ef | grep httpd
   ```

   The output should list a number of processes.

2. Check that the HTTP Server is registered to listen on port 80 and is therefore ready to handle requests:

   ```
   # netstat -an | grep LISTEN | grep 80
   ```

3. Check that the HTTP Administration Server is registered to listen on port 8008 and is therefore ready to handle requests:

   ```
   # netstat -an | grep LISTEN | grep 8008
   ```

Checking request handling by HTTP Server

To check IBM HTTP Server request handling, using a Web browser, request the following URL representing the IBM HTTP Server home page:

```
http://<hostname.domain.com>/
```

If the IBM HTTP Server has been installed and configured correctly, you should get a Welcome to IBM HTTP Server window.

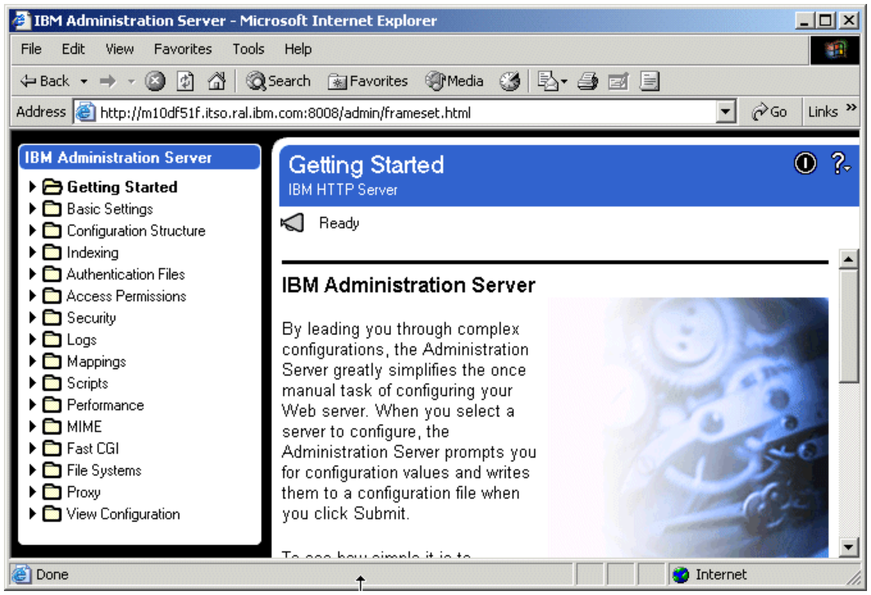
Checking request handling by HTTP administration server

To check HTTP administration server request handling, perform the following steps:

1. Using a Web browser, request the following URL representing the HTTP Administration Server home page:

   ```
   http://<hostname.domain.com>:8008/
   ```

2. Enter the user name and password specified in “Creating the HTTP server admin account” on page 172. The window shown in Figure 6-2 on page 175 will be displayed if the HTTP administration server has been installed and configured correctly.



By leading you through complex configurations, the Administration Server greatly simplifies the once manual task of configuring your Web server. When you select a server to configure, the Administration Server prompts you for configuration values and writes them to a configuration file when you click Submit.

Figure 6-2 Home page request handled by HTTP Administration Server

6.3.4 Verifying the plug-in installation

To verify that the Web server plug-in was successfully installed, open the httpd.conf configuration file. At the bottom of the file you should see the following entry:

```
LoadModule ibm_app_server_http_module
/usr/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /usr/WebSphere/AppServer/config/cells/plugin-cfg.xml
```

You will not be able to verify that the plug-in is functioning correctly until you have installed WebSphere Application Server and generated the Web server plug-in configuration.

Note the designated location of the plug-in configuration specified in httpd.conf, in this case, /usr/WebSphere/AppServer/config/cells/plugin-cfg.xml. This is where the plug-in will expect to find its configuration file. When you generate the Web server plug-in configuration from WebSphere Application Server, you will need to copy the generated file to this location.

Generating the Web server plug-in: Before you can use the Web server plug-in with a WebSphere Application Server installation, you will need to generate the plug-in configuration file and move it to the proper location on the Web server. Since this obviously cannot be done until WebSphere Application Server has been installed, this topic is covered later in 6.4.6, "Generating the Web server plug-in installation" on page 186.

6.4 Installing WebSphere Application Server

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server V5.1 for AIX.

Note: If you have a Web server on this system and are planning to now install the Web server plug-in, you should stop the Web server. As part of the Web server plug-in installation, updates will be made to the Web server configuration file. To stop the IBM HTTP Server, issue the following commands:

```
# cd <IHS_HOME>/bin
# ./apachectl stop
```

Space requirements: WebSphere Application Server installs in the /usr file system by default. The current space requirements for the features are:

- Application Server: 155.5 MB
- Administration: 88 MB
- Ant and deployment tools: 23.6 MB
- Embedded messaging: 71.8 MB
- Performance and Analysis Tools: 16.9 MB
- Javadocs: 10.3 MB
- Embedded messaging client: 10 MB

6.4.1 Installation

Embedded messaging users and groups: If you will be installing the embedded messaging features and have not created the users and groups required as outlined in 4.6.4, "Embedded messaging considerations (UNIX systems)" on page 128, do so now.

To install WebSphere Application Server using the GUI installer interface, complete the following steps:

1. Log in as root.

2. Start a terminal session.
3. Load the IBM WebSphere Application Server V5.1 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

4. Change the directory to /mnt.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set by the following commands:

   ```
   export DISPLAY=<IP address of host>:0.0
   export TERM=vt100
   ```

6. Run the LaunchPad.sh installation script:

   ```
   # ./LaunchPad.sh
   ```

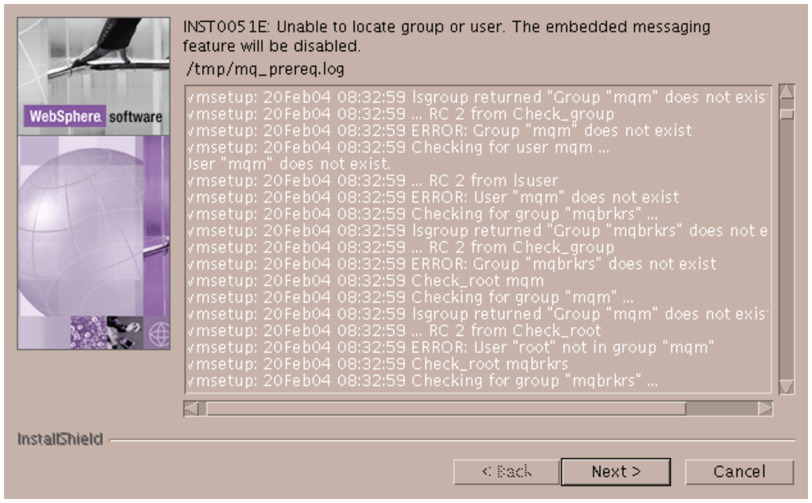
7. Select a language for the LaunchPad and click **OK**.
8. Review the *Readme* and *Installation Guide*.
9. Click **Install the product**.
10. Select a language to be used for the installation and click **OK**.
11. On the Welcome window, click **Next** to continue.
12. The software license agreement window will appear. Read the agreement, and accept the terms. Click **Next** to continue.
13. If you have previously installed WebSphere components (for example, the IBM HTTP Server, Web server plug-in, or Deployment Manager), you will be given the opportunity to add additional features to the current install or to install a new copy. You will also be given the opportunity to select alternate ports.
14. The installation will verify that your system has the required prerequisites.

   Note: If you are missing prereqs, you will be given a list of what is missing. You should stop at this point and bring the system up to the required level. However, if you choose to continue, the installation wizard will allow you to.

   Once the wizard confirms that all prerequisites have been met, or you elect to ignore the missing prereqs, the wizard will continue.

15. Select the **Custom** radio button and **Next** to continue.
16. Select the features to install. Click **Next**.
17. The wizard will perform a test install. If there are any problems, you will see a window ("Install check" on page 178) with messages indicating what the problem is. This process includes checking for the proper groups and users required for the embedded messaging feature. If the messages indicate that

the install will not be successful, examine them to determine what actions to take before proceeding.

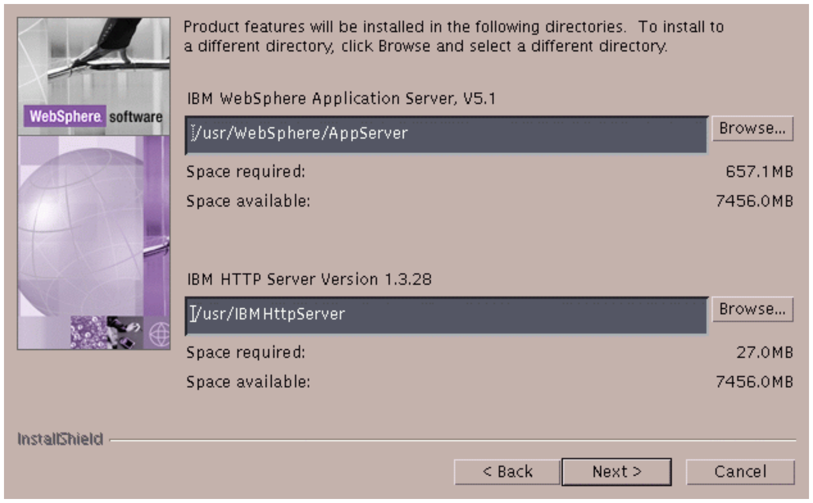

INST0051E: Unable to locate group or user. The embedded messaging feature will be disabled.

/tmp/mq_prereq.log

```
vmsetup: 20Feb04 08:32:59 lsgroup returned "Group "mqm" does not exis
vmsetup: 20Feb04 08:32:59 ... RC 2 from Check_group
vmsetup: 20Feb04 08:32:59 ERROR: Group "mqm" does not exist
vmsetup: 20Feb04 08:32:59 Checking for user mqm ...
Jser "mqm" does not exist.
vmsetup: 20Feb04 08:32:59 ... RC 2 from lsuser
vmsetup: 20Feb04 08:32:59 ERROR: User "mqm" does not exist
vmsetup: 20Feb04 08:32:59 Checking for group "mqbrkrs" ...
vmsetup: 20Feb04 08:32:59 lsgroup returned "Group "mqbrkrs" does not e
vmsetup: 20Feb04 08:32:59 ... RC 2 from Check_group
vmsetup: 20Feb04 08:32:59 ERROR: Group "mqbrkrs" does not exist
vmsetup: 20Feb04 08:32:59 Check_root mqm
vmsetup: 20Feb04 08:32:59 Checking for group "mqm" ...
vmsetup: 20Feb04 08:32:59 lsgroup returned "Group "mqm" does not exis
vmsetup: 20Feb04 08:32:59 ... RC 2 from Check_root
vmsetup: 20Feb04 08:32:59 ERROR: User "root" not in group "mqm"
vmsetup: 20Feb04 08:32:59 Check_root mqbrkrs
vmsetup: 20Feb04 08:32:59 Checking for group "mqbrkrs" ...
```

Figure 6-3 Install check

18.The next window (Figure 6-4) will enable the user to select the installation directories for IBM WebSphere Application Server and IBM HTTP Server. Note that you are prompted for the IBM HTTP Server location, even if you are not installing it. Simply accept the default. You will also see the space required versus the space available.


Product features will be installed in the following directories. To install to a different directory, click Browse and select a different directory.

IBM WebSphere Application Server, V5.1

/usr/WebSphere/AppServer

Space required: 657.1MB

Space available: 7456.0MB

IBM HTTP Server Version 1.3.28

/usr/IBMHttpServer

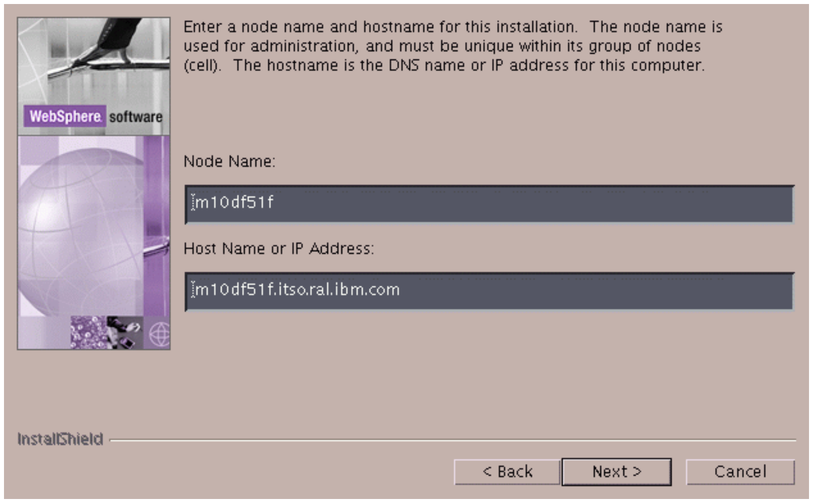
Space required: 27.0MB

Space available: 7456.0MB

Figure 6-4 Selecting the install directories

Verify that the install directories are correct and select **Next** to continue.

19. The next window (Figure 6-5) asks for the node name for this installation. The default is probably fine. However, keep in mind that the node name must be unique within a cell. The node name is a logical name, so although the default is the system host name, this is not mandatory.


Enter a node name and hostname for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer.

Node Name:

m10df51f

Host Name or IP Address:

m10df51f.itso.ral.ibm.com

Figure 6-5 Node name and host name

Click **Next** to continue.

20. The last window before installation begins shows a summary of all of the choices that were made in the custom installation. Select **Next** to begin the installation. A progress bar will indicate the status of installation.

21. Once the installation is complete, the installer will prompt the user to register. The user can choose to register the product immediately by checking the check box at the bottom of the window, or manually register at a later time by deselecting the check box.

22. Click **Next** and then **Finish**.

At the end of the installation, a new window, called the First Steps window, will open.

Changing port 9090

If you are running AIX 5.1 or AIX 5.2, it is likely that you will have a port conflict between the Web-based system manager of AIX and the WebSphere administrative console. To see if port 9090 is in use, issue the following command:

```
netstat -an | grep 9090
```

If port 9090 is in use (there is a line of output resulting from the command), you will need to change the port for the WebSphere administrative console.

Tip: As a temporary measure, you can disable the AIX system manager; this allows you to start WebSphere and use the administrative console to make changes to the WebSphere ports. To disable the system manager, enter:

```
/usr/websm/bin/wsmserver -disable
```

You can enable it using:

```
/usr/websm/bin/wsmserver -enable
```

To change the ports used by WebSphere, you will need to change the files listed in Table 6-3.

Table 6-3 Files to be changed

Setting	File
Virtual Host	<WAS_HOME>/config/cells/<cellname>/virtualhosts.xml
HTTP Transport	<WAS_HOME>/config/cells/<cellname>/nodes/<nodename>/servers/server1/server.xml

In the virtualhosts.xml file, find the following and change “9090” to a new port, for example, “9092”:

```
<host:VirtualHost xmi:id="VirtualHost_2" name="admin_host">
   <aliases xmi:id="HostAlias_4" hostname="*" port="9090"/>
   <aliases xmi:id="HostAlias_5" hostname="*" port="9043"/>
</host:VirtualHost>
```

In the server.xml file, find the following and change “9090” to the new port number you used above:

```
<transports xmi:type="applicationserver.webcontainer:HTTPTransport"
xmi:id="HTTPTransport_1" sslEnabled="false">
   <address xmi:id="EndPoint_1" host="" port="9090"/>
</transports>
```

In this example, the new definitions would look like the following:

```
<host:VirtualHost xmi:id="VirtualHost_2" name="admin_host">
   <aliases xmi:id="HostAlias_4" hostname="*" port="9092"/>
   <aliases xmi:id="HostAlias_5" hostname="*" port="9043"/>
```

```
</host:VirtualHost>


<transports xmi:type="applicationserver.webcontainer:HTTPTransport"
xmi:id="HTTPTransport_1" sslEnabled="false">
   <address xmi:id="EndPoint_1" host="" port="9092"/>
</transports>
```

Note: If the port number of the administrative console is changed, you will not be able to use the First Steps window to start it. This is not a problem, since you will normally start the console directly from a Web browser.

6.4.2 Verifying the WebSphere installation

In order to verify the installation of WebSphere Application Server, the following tasks should be completed:

1. Run the Installation Verification Test (IVT).
2. If the IVT is not successful, check the installation log for indications of what might have gone wrong.

IBM HTTP Server and Web server plug-in installation: If you install these features, refer to 6.3, “Installing the IBM HTTP Server and/or Web server plug-in” on page 169 for a discussion of post-installation activities.

Using the Installation Verification Test (IVT)

The IVT scans the product log files for errors and verifies the core functionality of the product. The IVT does the following:

1. Starts the application server (server1) if it has not been started yet and verifies its status.
2. Verifies the servlet engine (Web container) status.
3. Verifies the JSP status.
4. Verifies the EJB status.

The last step of the installation wizard starts the First Steps window. You can run the IVT from there. If you have closed this window, you can start it manually by issuing the following commands:

```
# cd <WAS_HOME>/bin
# ./firststeps.sh
```

To start the IVT from the First Steps window:

1. Click **Verify Installation**.

2. Monitor the window for messages indicating the success or failure of the tests. If the verification is successful, you will see the following messages:

```
IVTL0095I: defaulting to host m10df51f and port 9080
IVTL0010I: Connecting to the WebSphere Application Server m10df51f on port:
9080

IVTL0020I: Could not connect to Application Server, waiting for server to
start
IVTL0025I: Attempting to start the Application Server
osName = AIX
IVTL0030I: Running /usr/WebSphere/AppServer/bin/startServer.sh server1
>ADMU0116I: Tool information is being logged in file
>           /usr/WebSphere/AppServer/logs/server1/startServer.log
>ADMU3100I: Reading configuration for server: server1
>ADMU3200I: Server launched. Waiting for initialization status.
>ADMU3000I: Server server1 open for e-business; process id is 18252
IVTL0050I: Servlet Engine Verification Status - Passed
IVTL0055I: JSP Verification Status - Passed
IVTL0060I: EJB Verification Status - Passed
IVTL0070I: IVT Verification Succeeded
IVTL0080I: Installation Verification is complete
```

The IVT can also be run without using the First Steps window with the following commands:

```
# cd <WAS_HOME>/bin
# ./ivt.sh
```

Messages from the IVT test are stored in <WAS_HOME>/logs/ivt.log.

(Optional) Checking the installation log

If you find that the IVT does not complete correctly, you can check the log files listed in Table 6-4 under <WAS_HOME>/logs directory for installation errors.

Table 6-4 Installation log files

Component	File
WebSphere Application Server	log.txt
IBM HTTP Server	ihs_log.txt
Default Application	installDefaultApplication.log
Sample Application	installSamples.log
Administrative console	installAdminConsole.log
MDB Samples Application	installMessagingSamples.log

Component	File
Pet Store Application	installPetStore.log
Plants By WebSphere Application	installPlantsByWeb.log
Technology Samples Application	installTechSamples.log
IVT Application	installIVTApp.log

The installation wizard also creates the optional_log.txt file in the $TEMP directory. Check this file if necessary.

6.4.3 Starting and stopping server1

WebSphere Application Server is installed with one server ("server1") preconfigured. The application that provides the administrative console is installed on this server, as well as the sample applications. You can install additional applications on this server and you can create new servers.

Before accessing the administrative console or applications that are installed on the server you will need to start "server1". You have seen how to do this from the First Steps window, but after installation, it is unlikely that you will use the First Steps. For future reference, you can start or stop server1 by doing the following:

```
# cd <WAS_HOME>/bin
# ./startServer.sh server1
or
# ./stopServer.sh server1
```

When starting the server, you should see the following message if the start was successful:

```
WSVR0001I: Server server1 open for e-business
```

If the server did not start successfully, check SystemOut.log and SystemErr.log. They are found in the <WAS_HOME>/logs/server1 directory.

As a general practice, you may want to monitor the SystemOut.log. To do this, enter the following commands:

```
# cd <WAS_HOME>/logs/server1
# tail -f SystemOut.log
```

6.4.4 Accessing the administrative console

To access the administrative console:

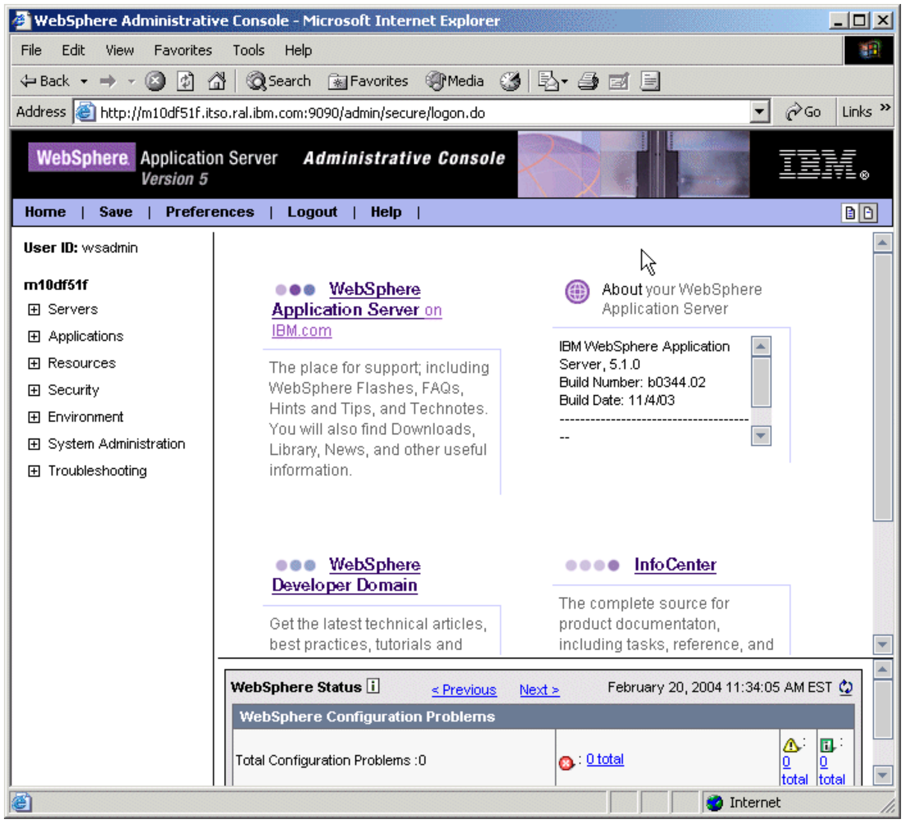
1. Using a Web browser, request the following URL:

   `http://<hostname>:<port>/admin`

   Where `<hostname>` is the host name of the WebSphere Application Server and <port> is the port number for the administrative console. By default, this is 9090, but if you changed it, use the new port number.

2. Enter a user ID to log in. Until you enable security, this user ID is used for tracking purposes only, so any user ID is acceptable. The user does not have to be a valid system user. In this example, *user* is used as the user ID.

The window shown in Figure 6-6 will appear in the browser.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://m10df51f.itso.ral.ibm.com:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: wsadmin
m10df51f
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
About your WebSphere Application Server
WebSphere Developer Domain
InfoCenter
WebSphere Status
< Previous
Next >
WebSphere Configuration Problems
Internet

The place for support; including WebSphere Flashes, FAQs, Hints and Tips, and Technotes. You will also find Downloads, Library, News, and other useful information.

IBM WebSphere Application Server, 5.1.0
Build Number: b0344.02
Build Date: 11/4/03

--

Get the latest technical articles, best practices, tutorials and

The complete source for product documentaton, including tasks, reference, and

February 20, 2004 11:34:05 AM EST

Total Configuration Problems :0 : 0 total : 0 total : 0 total

Figure 6-6 Admin console first page

6.4.5 Accessing applications on WebSphere

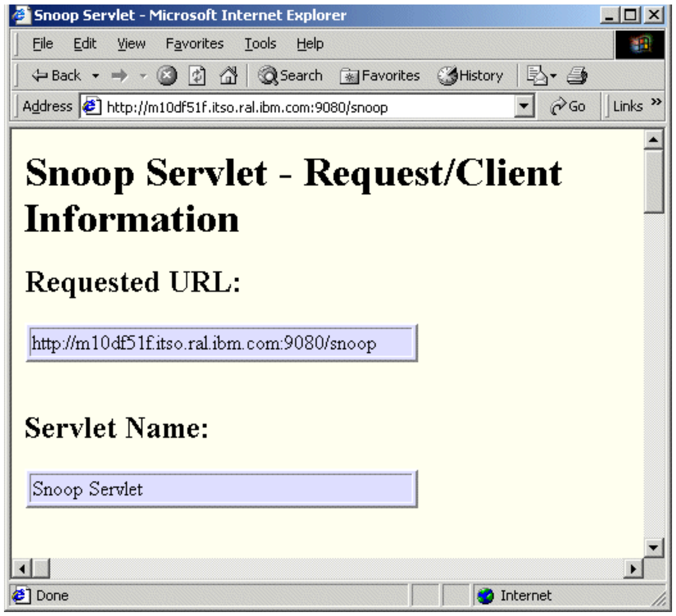
When you install WebSphere Application Server, you automatically have at least one application installed. The application, called DefaultApplication, exists simply to help you verify that the installation is working. You can access the servlets in DefaultApplication with the following URLs:

- `http://<WAS_hostname>:9080/snoop`
- `http://<WAS_hostname>:9080/hello`
- `http://<WAS_hostname>:9080/hitcount`

If you installed the sample applications, you can access them with the following URL:

```
http://<hostname>:9080/WSsamples
```

Note that in each case the URL specifically designates port 9080. This is the default HTTP transport port. If you changed this during installation, you will need to substitute the correct port number. Once you have set up a Web server and the Web server plug-in, you will be able to access these applications through the Web server and will not need to designate a port.


Snoop Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://m10df51f.itso.ral.ibm.com:9080/snoop
Go Links
Done
Internet

Snoop Servlet - Request/Client Information

Requested URL:

http://m10df51f.itso.ral.ibm.com:9080/snoop

Servlet Name:

Snoop Servlet

Figure 6-7 Snoop servlet

6.4.6 Generating the Web server plug-in installation

Network Deployment: If you are going to install Network Deployment and will be adding this WebSphere Application Server to a cell, you can skip this step and do it after the cell is set up.

To generate and verify the Web server plug-in, you will need to do the following:

1. Verify that the plug-in statements have been added to the Web server configuration file.
2. Generate the configuration file from WebSphere Application Server. Using the administrative console, expand **Environment** and select **Update Web Server Plugin**. Click **OK** to generate the configuration file.
3. Copy the configuration to the Web server system. The location you need to copy it to is specified in the WebSpherePluginConfig directive of the Web server configuration file. If you installed WebSphere Application Server and the IBM HTTP Server on the same system, the generated plug-in will be placed in the correct location automatically.
4. Test the connection to WebSphere through the Web server. To test the remote Web server connection the WebSphere, access the snoop servlet using the following URL:

   ```
   http://<Web server hostname>/snoop
   ```

6.4.7 Implementing security

Network Deployment: If you are going to install Network Deployment and will be adding this WebSphere Application Server to a cell, you can skip this step and do it after the cell is set up.

Security is an important part of any e-business environment. Designing and implementing the security structure is a topic of its own and is addressed in detail in *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

However, at this time, you should consider securing the administrative console to restrict who has access to the WebSphere configuration. The instructions to do this can be found in 8.5, “Securing the administrative console” on page 276.

6.5 Installing WebSphere Application Server Network Deployment

This section provides detailed instructions for installing, configuring, and verifying WebSphere Application Server Network Deployment for AIX.

Space requirements: WebSphere Application Server Network Deployment installs in the /usr file system by default. The current space requirements for the features are:

- Deployment Manager: 133.2 MB
- Web services: 23.4 MB
- Embedded messaging client: 10 MB

6.5.1 Installing Network Deployment

To install WebSphere Application Server Network Deployment using the GUI installer interface, complete the following steps:

1. Log in as root.
2. Start a terminal session.
3. Load the IBM WebSphere Application Server V5.0 Network Deployment CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

4. Change the directory to /mnt.
5. Ensure the DISPLAY and TERM environment variables are properly set. They can be set by the following commands:

   ```
   export DISPLAY=<IP address of host>:0.0
   export TERM=vt100
   ```

6. Run the LaunchPad.sh installation script:

   ```
   # ./LaunchPad.sh
   ```

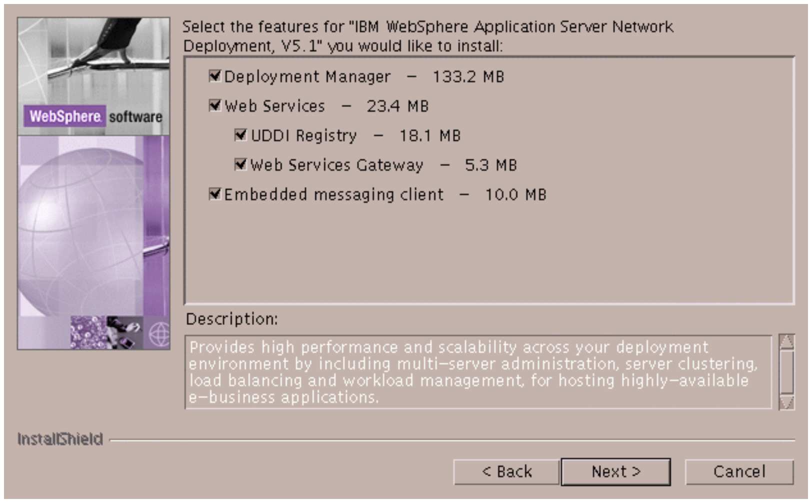
7. Select a language for the LaunchPad and click **OK**.
8. Review the *Readme* and *Installation Guide*.
9. Click **Install the product**.
10. Select the language to be used for the installation, select **OK**.
11. Select **Next** on the Welcome window to continue.
12. The License Agreement will be displayed. Read the License Agreement and if you agree to its terms, accept them and select **Next** to continue.

13. If you have previously installed WebSphere components (for example, the IBM HTTP Server, Web server plug-in, or WebSphere Application Server), you will be given the opportunity to add additional features to the current install or to install a new copy. You will also be given the opportunity to select alternate ports.
14. The installation will verify that your system has the required prerequisites.

> **Note:** If you are missing prerequisites, you will be given a list of what is missing. You should stop at this point and bring the system up to the required level. However, if you choose to continue, the installation wizard will allow you to.

Once the wizard confirms that all prerequisites have been met, or you elect to ignore the missing prerequisites, the wizard will continue.

15. Select the **Custom** radio button and **Next** to continue.
16. Select the features to install. Click **Next**.



Select the features for "IBM WebSphere Application Server Network Deployment, V5.1" you would like to install:

- ☑ Deployment Manager – 133.2 MB
- ☑ Web Services – 23.4 MB
  - ☑ UDDI Registry – 18.1 MB
  - ☑ Web Services Gateway – 5.3 MB
- ☑ Embedded messaging client – 10.0 MB

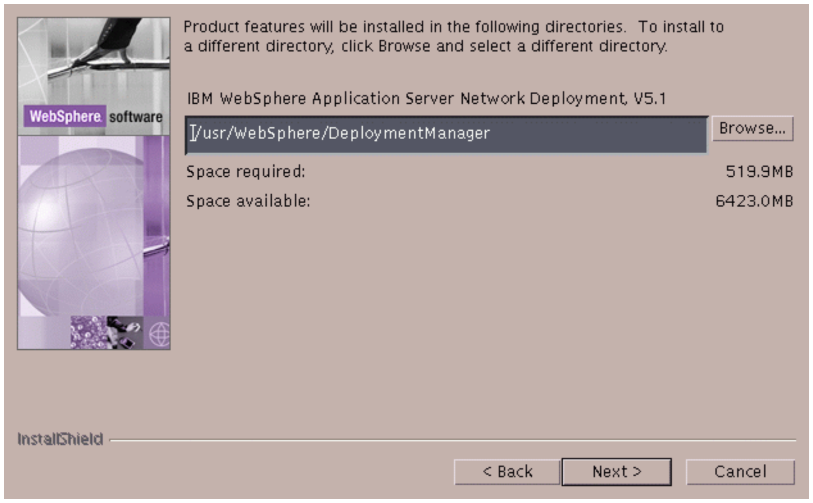
Description:

Provides high performance and scalability across your deployment environment by including multi–server administration, server clustering, load balancing and workload management, for hosting highly–available e–business applications.

Figure 6-8 Select the features window (ND)

17. The next window will prompt for a directory in which to install the Deployment Manager. Accept the default or change it to meet your installation standards.

Note that you can see the space requirements for the install and the available space on your system.


Product features will be installed in the following directories. To install to a different directory, click Browse and select a different directory.

IBM WebSphere Application Server Network Deployment, V5.1

/usr/WebSphere/DeploymentManager

Space required: 519.9MB

Space available: 6423.0MB

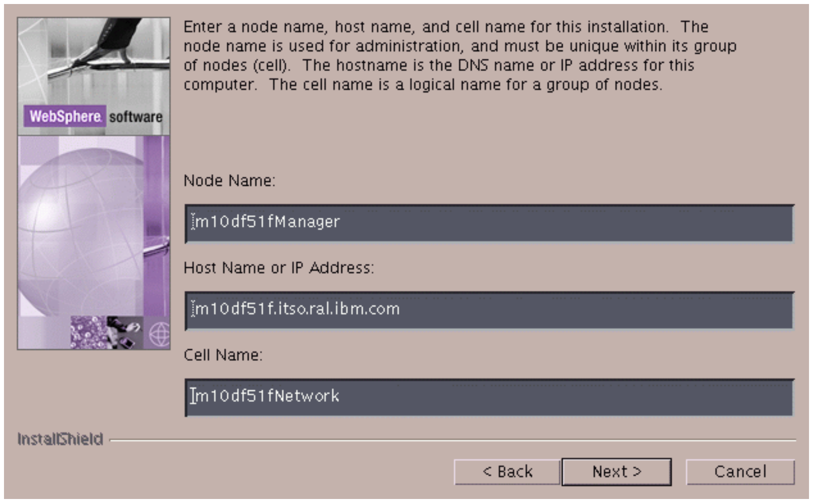
Figure 6-9 Install directory and space requirements

Click **Next**.

18. The next window will prompt for the Network Deployment node name, host name, and cell name for your installation.

    The node name must be unique in the cell. If you select a name that is not unique, the install will continue, but you will have problems later when you attempt to add the node to the cell. The defaults use the host name of your machine as a basis.

    Select **Next** to continue.


Enter a node name, host name, and cell name for this installation. The node name is used for administration, and must be unique within its group of nodes (cell). The hostname is the DNS name or IP address for this computer. The cell name is a logical name for a group of nodes.

Node Name: m10df51fManager

Host Name or IP Address: m10df51f.itso.ral.ibm.com

Cell Name: m10df51fNetwork

Figure 6-10 Node name, host name and cell name window (ND)

19. The last window before installation is the Summary window. The window will detail the sections that were made during the install. Verify that these selections are correct, and select **Next** to begin copying files to the machine.
20. You will be given a chance to register the product. After doing so, click **Finish** to exit the install.

6.5.2 Verifying the Deployment Manager installation

As the last step of the installation, the First Steps window is opened. From this window you can view the InfoCenter, start and stop the Deployment Manager, run the installation verification tests, and access the administrative console.

If you want to start this window in the future, you can do so with the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./firststeps.sh
```

Changing port 9090

Both WebSphere Application Server and Network Deployment use port 9090 for the administrative console function by default. On AIX V5.1 or AIX V5.2, it is highly likely that you will have a conflict with the Web-based system manager of AIX.

To see if port 9090 is in use, issue the following command:

```
netstat -an | grep 9090
```

If port 9090 is in use (there is a line of output resulting from the command), you will need to change the port for the WebSphere administrative console.

Tip: As a temporary measure, you can disable the AIX system manager; this allows you to start WebSphere and use the administrative console to make changes to the WebSphere ports. To disable the system manager, use:

```
/usr/websm/bin/wsmserver -disable
```

You can enable it using:

```
/usr/websm/bin/wsmserver -enable
```

To change the port number for WebSphere, you will need to change the files listed in Table 6-5.

Table 6-5 Files to be changed

Setting	File
Virtual Host	<WAS_ND_HOME>/config/cells/<cellname>/virtualhosts.xml
HTTP Transport	<WAS_ND_HOME>/config/cells/<cellname>/nodes/<nodename>/ servers/dmgr/server.xml

To change the virtual host setting, open the virtualhosts.xml file. Find `9090` and change it to a new port. In this example, the port number is changed from 9090 to 9092:

```
<host:VirtualHost xmi:id="VirtualHost_2" name="admin_host">
   <aliases xmi:id="HostAlias_4" hostname="*" port="9092"/>
   <aliases xmi:id="HostAlias_5" hostname="*" port="9043"/>
</host:VirtualHost>
```

To change the HTTP transport setting, open the server.xml file. Find `9090` and change it to the same new port number you used in the virtualhosts.xml file. In this example, the port number is changed from 9090 to 9092:

```
<transports xmi:type="applicationserver.webcontainer:HTTPTransport"
xmi:id="HTTPTransport_1" sslEnabled="false">
   <address xmi:id="EndPoint_1" host="" port="9092"/>
</transports>
```

Note: If the port number of the administrative console is changed, you will not be able to use the First Steps window to start it. This is not a problem, since you will normally start the console directly from a Web browser.

Using the Installation Verification Test (IVT)

The IVT scans the product log files for errors and verifies the core functionality of the product by starting the Deployment Manager and verifying its status.

The last step of the installation wizard starts the First Steps window. You can run the IVT from there. If you have closed this window, you can start it manually by issuing the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./firststeps.sh
```

To start the IVT from the First Steps window:

1. Click **Verify Installation**.
2. Monitor the window for messages indicating the success or failure of the tests. If the verification is successful you will see the following messages:

```
IVTL0095I: defaulting to host m10df51f and port 9091
IVTL0010I: Connecting to the WebSphere Application Server m10df51f on port:
9091

IVTL0020I: Could not connect to Application Server, waiting for server to
start
IVTL0025I: Attempting to start the Application Server
osName = AIX
IVTL0030I: Running /usr/WebSphere/DeploymentManager/bin/startManager.sh
>ADMU0116I: Tool information is being logged in file
>           /usr/WebSphere/DeploymentManager/logs/dmgr/startServer.log
>ADMU3100I: Reading configuration for server: dmgr
>ADMU3200I: Server launched. Waiting for initialization status.
>ADMU3000I: Server dmgr open for e-business; process id is 24258
IVTL0070I: IVT Verification Succeeded

IVTL0080I: Installation Verification is complete
```

The IVT can also be run without using the First Steps window with the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./ivt.sh
```

Messages from the IVT test are stored in <WAS_ND_HOME>/logs/ivt.log.

Checking the installation log

If the IVT is not successful, check the installation log files under <WAS_ND_HOME>/logs directory for errors.

Table 6-6 Installation log files (ND)

Component	File
Network Deployment	log.txt
Administrative Console	installAdminconsole.log
File Transfer Application	installFiletransfer.log

Starting the Deployment Manager

The Deployment Manager must be started before accessing the administrative console. As mentioned, you can do this from the First Steps window, but after installation, it is unlikely that you will use the First Steps. For future reference, you can start or stop the Deployment Manager with the following commands:

```
# cd <WAS_ND_HOME>/bin
# ./startManager.sh
```

If the start is successful, you should see the following message:

```
WSVR0001I: Server dmgr open for e-business
```

If the Deployment Manager fails to start, check the SystemOut.log and SystemErr.log log files found in the <WAS_ND_HOME>/logs/dmgr directory.

If you want to monitor the Deployment Manager system messages, issue the following commands:

```
# cd <WAS_ND_HOME>/logs
# tail -f SystemOut.log
```

Opening the administrative console

To access the administrative console:

1. Using a Web browser, request the following URL:

   ```
   http://<hostname>:<port>/admin
   ```

   Where `<hostname>` is the host name of the WebSphere Application Server and `<port>` is the port number for the administrative console. By default, this is 9090, but if you changed it, use the new port number.

2. Enter a user ID to log in. Until you enable security, this user ID is used for tracking purposes only, so any user ID is acceptable. The user does not have to be a valid system user. In this example, *user* is used as the user ID.

6.5.3 Adding a WebSphere node into an existing cell

You have just been through the process of installing a Network Deployment Deployment Manager. This gives you a cell framework, but as of yet, there are no nodes in the cell. Presumably you have also installed one or more WebSphere Application Servers. If not, you should do this now.

In order for a node to be managed by the Deployment Manager, it must be added to the cell.

1. On the Deployment Manager node, start the Deployment Manager.

   ```
   cd <WAS_ND_HOME>/bin
   ./startManager.sh
   ```

2. On the WebSphere Application Server node, stop all the servers. To do this, open a command prompt and check the status of the servers with the following commands:

   ```
   cd <WAS_HOME>/bin
   ./serverStatus.sh -all
   ```

   If this is a new install, you should only have one server, server1. If server1 is started, stop it with the following command:

   ```
   ./stopServer.sh server1
   ```

3. On the WebSphere Application Server node, add the node to the cell with the following command:

   ```
   ./addNode.sh <DM_hostname> 8879 -includeapps
   ```

   Note that the addNode command is run from the application server side, not the Network Deployment side. The `<DM_hostname>` is the host name of the Network Deployment machine and the port (8879) is the SOAP connector port for the Deployment Manager.

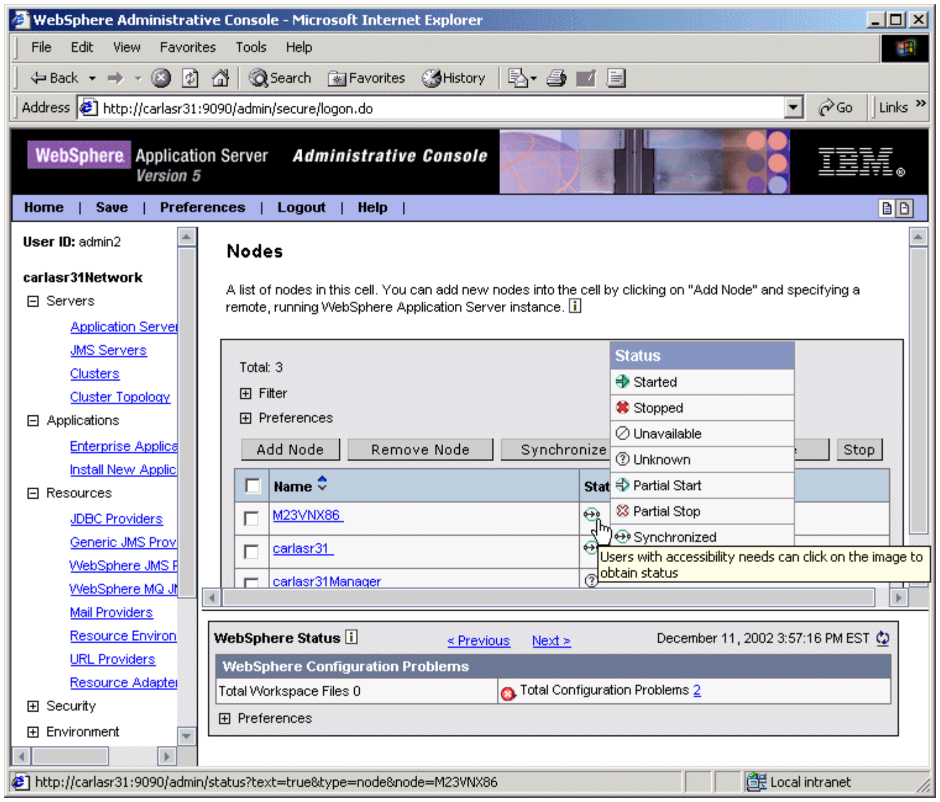
   Finding the SOAP connector port: If you need to look up the SOAP connector port for the Deployment Manager, do the following:

   1. Open the administrative console for the Deployment Manager.
   2. Select **System Administration -> Deployment Manager**.
   3. Select **End Points** in the Additional Properties table.
   4. Select **SOAP_CONNECTOR_ADDRESS.**

   When the process is done, you should get the following message:

   ```
   Node nodename has been successfully federated.
   ```

You can see the new node in the Network Deployment administrative console by expanding System Administration and clicking **Nodes**. Check the status of the new node and make sure it is synchronized. If not, select the new node and click **Synchronize**.



Nodes

A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote, running WebSphere Application Server instance.

Total: 3

	Name	Stat
	M23VNX86	
	carlasr31	
	carlasr31Manager	

Users with accessibility needs can click on the image to obtain status

WebSphere Status

December 11, 2002 3:57:16 PM EST

WebSphere Configuration Problems

Total Workspace Files 0	Total Configuration Problems 2

Figure 6-11 Display the nodes in the cell

Starting the node agent: After adding a node to a cell, a node agent server is required to be running on the node in order for the Deployment Manager to communicate with it. The node agent is started automatically during the addnode process but subsequently must be started manually by entering the following from a command prompt on the WebSphere Application Server machine:

```
cd <WAS_HOME>/bin
./startnode.sh
```

Starting server1: Once the node agent is started, you can start the server from the Deployment Manager administrative console. Expand **Servers** and select **Application Servers** to see server1. Select (check the box) server1 and click **Start**.

6.6 Installing WebSphere Application Server - silent mode

This section describes how to install WebSphere Application Server using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

If you are using IBM HTTP Server as your Web server, you will install it at the same time and onto the same node as you install WebSphere Application Server. If you are using another supported Web server with WebSphere Application Server, you have already installed it onto the same node as WebSphere Application Server.

Planning and post-installation activities for silent installs remain the same as with a GUI installation and will not be addressed here.

Note: IBM HTTP Server is supplied with WebSphere Application Server. If you plan to use a different Web server, you must purchase it and install it separately. It is recommended that the Web server be installed before WebSphere Application Server.

6.6.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server as a template for creating a customized response file.

With the default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.4.1
- IBM HTTP Server 1.3.28
- IBM WebSphere Application Server 5.1
- Embedded messaging
- Performance and analysis tools
- WebSphere Application Server application samples
- Documentation in U.S. English

Note: All products except IBM HTTP Server are installed into the directory /usr/WebSphere/AppServer. IBM HTTP Server is installed into the directory /usr/IBMHttpServer. In addition, WebSphere Application Server is configured for use with IBM HTTP Server when you use the default response file.

6.6.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components of WebSphere Application Server or to install the products in a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

The following lines are an example of which lines you may want to update:

```
-P wasBean.installLocation="/usr/WebSphere/AppServer"
-P ihsFeatureBean.installLocation="/usr/IBMHttpServer"
-W nodeNameBean.nodeName="m10df51f"
-W nodeNameBean.hostName="9.24.104.21"
-W defaultIHSConfigFileLocationBean.value="/usr/IBMHttpServer/conf/httpd.conf"
```

In this example, the following items are specified in the response file respectively.

- WebSphere installation directory
- IBM HTTP Server installation directory
- Node name
- Host name or IP address
- Configuration file of IBM HTTP Server

Important: If you have a port conflict with port 9090, you can change this port in the response file, saving the trouble of changing it after installation.

6.6.3 Performing a silent installation

Perform the following steps to create a customized response file (if desired) to install WebSphere Application Server. These instructions assume that the installation is being performed from the product CD-ROM:

1. Ensure that you are logged into the machine with superuser (root) privileges.
2. If a Web server is running on your system, stop the Web server. If you plan to install IBM HTTP Server 1.3.26 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.26 on your system, you must uninstall it for the WebSphere Application Server installation program to install IBM HTTP Server 1.3.26.
3. Load the IBM WebSphere Application Server V5.0 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

4. Navigate to the /mnt directory.
5. Ensure that you are in the /mnt directory and create a copy of the default response file by using the **cp** command, as follows:

   ```
   # cp responsefile.txt <new_responsefile.txt>
   ```

   In this command, `<new_responsefile.txt>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my`_responsefile.txt).
6. To create a customized response file, perform the following steps:
   a. Use a text editor to open your copy of the default response file, <new_responsefile.txt>.
   b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation. See “Customized response file” on page 197 for a list of minimum changes to the response file.
   c. Save the changes that you have made to the customized response file.
7. Run the installation script by using the following commands.

   ```
   # ./install.sh -options <new_responsefile.txt>
   ```

   The install.sh script uses the response file to install the components and options that you have selected. The variable <new_responsefile.txt> represents the full path name of the copy of the default response file or the customized response file that you have created (for example, /tmp/new_responsefile.txt).
8. After installation is completed, refer to the log file named log.txt located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the <WAS_HOME>/logs directory.

9. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **umount** command, as follows:

   ```
   # umount /mnt
   ```

6.7 Installing Network Deployment - silent mode

This section describes how to install WebSphere Application Server Network Deployment using the non-interactive, or silent, mode. To complete a silent installation, you will create a customized response file from the default one, then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

These instructions assume that your machine has sufficient memory and disk space for your installation. Planning and post-installation activities for silent installs are the same as with a GUI installation and will not be addressed here.

Important: If you have a port conflict with port 9090, you can change this port in the response file, saving the trouble of changing it after installation.

6.7.1 Default response file

A default response file, named responsefile.txt, is supplied with WebSphere Application Server Network Deployment. You can use this default response file to install WebSphere Application Server as a template for creating a customized response file.

With the default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.4.1
- Deployment Manager
- Embedded messaging
- Web services

6.7.2 Customized response file

The default response file is used as a template for creating a customized response file. The default response file can be edited to select the components of WebSphere Application Server Network Deployment or to install the products in a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

The following lines are an example of which lines should be updated at least for the basic installation.

```
-P wasBean.installLocation="/usr/WebSphere/DeploymentManager"
-W nodeNameBean.nodeName="m10df51fManager"
-W nodeNameBean.cellName="m10df51fNetwork"
-W nodeNameBean.hostName="m10df51f"
```

In the example, the following items are specified in the response file respectively:

- Network Deployment installation directory
- Node name
- Cell name
- Host name or IP address

Important: All values should be enclosed in double quotes (““).

6.7.3 Performing a silent installation

Perform the following steps to create a customized response file (if desired) to install WebSphere Application Server Network Deployment. These instructions assume that the installation is being performed from the product CD-ROM:

1. Ensure that you are logged into the machine with superuser (root) privileges.
2. Load the IBM WebSphere Application Server Network Deployment V5.0 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   # mount -r -v cdrfs /dev/cd0 /mnt
   ```

3. Navigate to the /mnt directory.
4. Ensure that you are in the /mnt directory and create a copy of the default response file by using the **cp** command, as follows:

   ```
   # cp responsefile.txt <new_responsefile.txt>
   ```

   In this command, `<new_responsefile.txt>` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my_`responsefile.txt).
5. To create a customized response file, perform the following steps:
   a. Use a text editor to open your copy of the default response file, <new_responsefile.txt>.
   b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server Network Deployment installation. See “Customized response file” on page 199 for a list of minimum changes to the response file.
   c. Save the changes that you have made to the customized response file.

6. Run the installation script by using the following commands. The install.sh script uses the response file to install the components and options that you have selected. The variable `<new_responsefile.txt>` represents the full path name of the copy of the default response file or the customized response file that you have created (for example, /tmp/new_responsefile.txt).

   ```
   # ./install.sh -options <new_responsefile.txt>
   ```

7. After installation is completed, see the log.txt file located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the directory <WAS_ND_HOME>/logs.
8. Unmount the CD-ROM before removing it from the CD-ROM drive by using the **umount** command, as follows:

   ```
   # umount /mnt
   ```

Part 3

Configuring WebSphere

This part takes you through the process of configuring WebSphere Application Server. It introduces the concepts required for the administration of WebSphere and then goes into more detail about the tasks required.

7

Administration overview

This chapter describes in detail the system management functionality of IBM WebSphere Application Server Network Deployment. We cover:

- Key features of IBM WebSphere Application Server system management
- System management topology
- Distributed administration
- Configuration and application repositories
- Application management
- System management tools
- Common system management tasks

7.1 Key features

The system management functionality of IBM WebSphere Application Server Network Deployment has a number of key features:

- Application servers have less reliance on a central repository or administration server for basic functions and bring-up.
- No administrative repository database is required. Instead, the configuration is stored in XML format files.
- Administration client programs are used to modify configuration settings:
  - Web-based administration (administrative console).
  - WebSphere scripting (wsadmin).
- Each managed process, node agent, and Deployment Manager starts with its own configuration file.

  Note: Compare this with WebSphere Application Server V4, where the single admin.config configuration file was used to start all managed processes.

- WebSphere managed processes are organized into groups. All managed processes configured and running on the same node must enroll with the local node agent. All node agent processes configured in the same cell must enroll with the Deployment Manager.
- The use of Java Management Extensions (JMX). This provides a number of benefits:
  - Management of WebSphere using multiple protocols. For example:
    - SOAP, RMI/IIOP (current)
    - HTTP, JMS, SNMP (future)
  - Easy to add or extend administrative functions by adding custom management beans (MBeans).
    - MBeans are defined in an XML file.
    - Allows developers to use JMX to manage their own applications.
  - The management interface is open and easy to integrate with third-party management tools.
- Resources within WebSphere managed processes are represented by individual management interfaces implemented as JMX MBeans. These managed resources are accessible from other processes through one of the WebSphere AdminService proxies.

Note: Third-party and custom applications can also implement their management interfaces as JMX MBeans, which can be deployed and executed in the application server runtime.

7.2 System management tools

The IBM WebSphere Application Server administration tools are used for system management for the entire distributed topology:

- WebSphere administrative console
- Command-line operational tools
- WebSphere scripting using wsadmin
- Java-based JMX APIs that can be accessed directly by custom Java applications.

Although any of these interfaces can be used to configure system management functions, use of the administrative console is preferred because it validates any changes made to the configuration.

7.2.1 WebSphere administrative console

The administrative console runs as a J2EE 1.3 Web application within a managed server process on a node. The location and configuration of the console differs depending upon whether the base or Network Deployment environment is in use.

Base

In a base installation the administrative console is installed and accessed on the default application server (server1).

Note: New application servers created using the admin tools do not have the administrative console installed. However, the administrative console application can be installed manually into each application server if so desired.

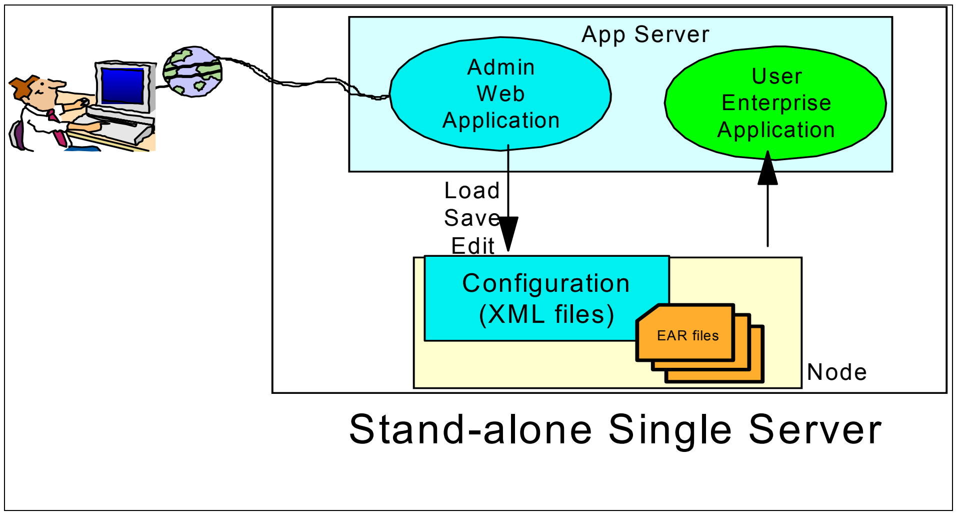

App Server
Admin Web Application
User Enterprise Application
Load
Save
Edit
Configuration (XML files)
EAR files
Node
Stand-alone Single Server

Figure 7-1 Stand-alone server WebSphere administrative console

Network Deployment

In the Network Deployment configuration, the administrative console is hosted on the Deployment Manager process. The act of adding a node to a cell results in the administrative console being removed from any application server. However, the administrative console application can be installed manually into one or more application servers, for example, in a very large cell configuration where some local Web-based “tweaking” of application server settings is desired.

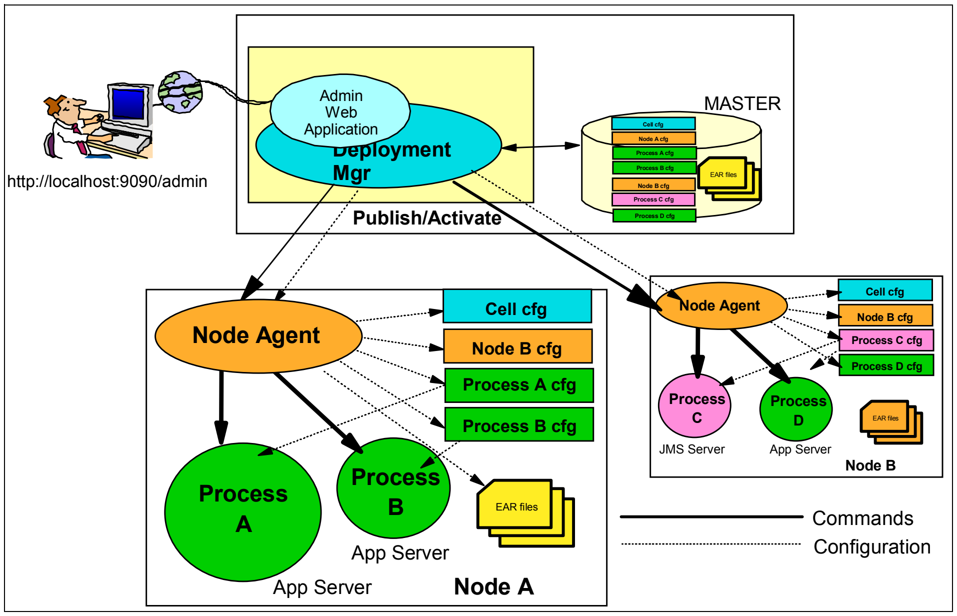


Figure 7-2 Deployment Manager WebSphere administrative console

For detailed information regarding the features and use of the WebSphere administrative console, see Chapter 8, “WebSphere administration basics” on page 257.

7.2.2 Command-line operational tools

IBM WebSphere Application Server provides command-line tools for operational administration of the runtime environment. These commands are discussed in Appendix A, “Command-line tools” on page 833.

7.3 Topology

The components of the IBM WebSphere Application Server Network Deployment System Management topology are summarized in Figure 7-3 on page 210.

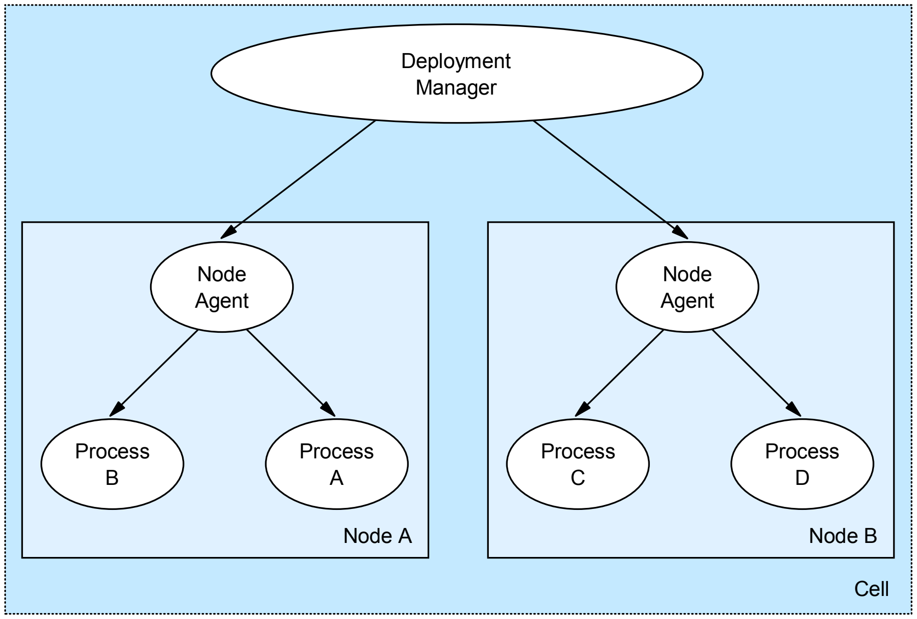


Figure 7-3 System management topology

7.3.1 Definitions

The major new terms used include:

- Cell

  A cell is an aggregation of nodes. A Deployment Manager controls and communicates with all node agents that are part of the cell, providing centralized system management and administration.

Notes:

1. Unlike previous versions of WebSphere, a WebSphere Application Server V5 administration grouping (cell) can contain nodes that are hosted by machines running different operating systems. As an example, consider the following supported three-machine topology:
   - Deployment Manager - hosted on machine 1 (AIX)
   - Node A - hosted on machine 2 (Windows 2000)
   - Node B - hosted on machine 3 (Linux)
   - Node C - hosted on machine 2 (Windows 2000)
2. IBM WebSphere Application Server Network Deployment does not provide support for the management of multiple cells. Each cell is a separate administrative domain and must be managed separately. A future edition of WebSphere Application Server V5 may allow multiple cells to be managed together.

- Node

  A node is a set of managed servers on a physical machine in a topology composed of one or more machines. A node contains an IBM WebSphere Application Server installation and is managed by a node agent process on a single machine.

  A node cannot span machines, but a single machine can host multiple nodes.

- Deployment Manager

  The process responsible for the management and control of the cell configuration and application data repository.

Notes:

1. The Network Deployment installation (containing the Deployment Manager) can be located on the same machine as one (or more) node installations. This would be a common topology for single machine development and testing environments.
2. In many production topologies it is recommended that the Deployment Manager installation be placed on a separate dedicated machine.

- Node agent

  The process responsible for controlling the managed processes of a node.

- Managed server

  A managed server is a single IBM WebSphere Application Server process, running in its own JVM. All operating system processes that are components of the WebSphere product are called managed servers. This means that the processes all participate in the administrative domain (cell). WebSphere support for JMX is embedded in all managed servers and these servers are available to receive administration commands and to output administration information about the state of the managed resources within the servers.

7.3.2 Operation

Network Deployment can be installed on any machine in the network to create a Deployment Manager cell. It does not require the base WebSphere Application Server to be installed on the same machine, although that is possible.

The `addNode` command can be run (on a base installation) to add a stand-alone base instance to the Deployment Manager cell. The Deployment Manager will create configuration files for each node that has been added to its cell and assumes responsibility for the configuration of all servers on the node.

The successful operation of a Network Deployment topology requires that each process run a specific set of system management services. In addition, some applications are used specifically to support the system management function, and therefore must be installed on specific servers. See 7.4.1, “Administration layers” on page 212 for details.

7.4 Distributed administration

Network Deployment allows multiple WebSphere Application Server nodes to be managed from a single central location. Distributed administration requires Network Deployment to be installed on a single machine in the environment.

7.4.1 Administration layers

The distributed administration of components is brought about by three tiers (layers) of administration services, as shown in Figure 7-4 on page 213.

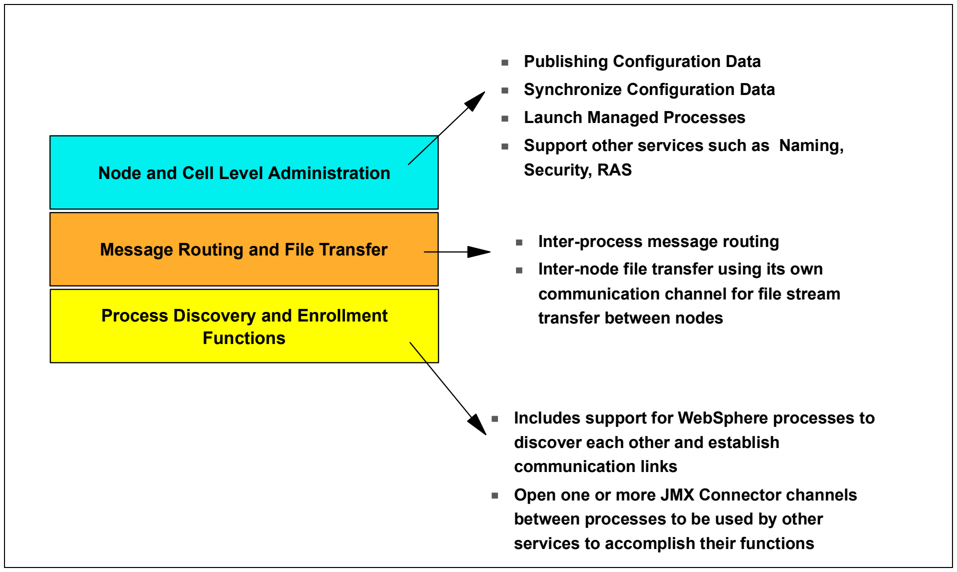

Node and Cell Level Administration
Message Routing and File Transfer
Process Discovery and Enrollment Functions
Publishing Configuration Data
Synchronize Configuration Data
Launch Managed Processes
Support other services such as Naming, Security, RAS
Inter-process message routing
Inter-node file transfer using its own communication channel for file stream transfer between nodes
Includes support for WebSphere processes to discover each other and establish communication links
Open one or more JMX Connector channels between processes to be used by other services to accomplish their functions

Figure 7-4 Layers of distributed administration services

Between these tiers, communication is used to distribute configuration and application data updates from the Deployment Manager to the node agents, and in turn to the server instances, as shown in Figure 7-5.

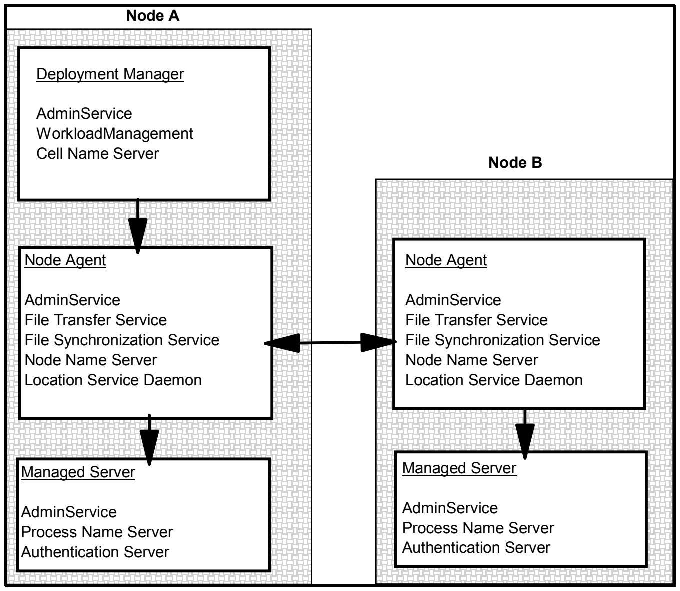


Figure 7-5 Distributed administration communication flow

The routing of administration messages between components makes use of the JMX ObjectName that identifies the target managed resource within the administrative cell. The ObjectName contains all of the information necessary to route a request targeted at the resource, to the appropriate node where the resource is executing.

An example is shown in Figure 7-6, where an operation on node “Y” invokes a management method on a management bean (MBean) located on another node, “X”.

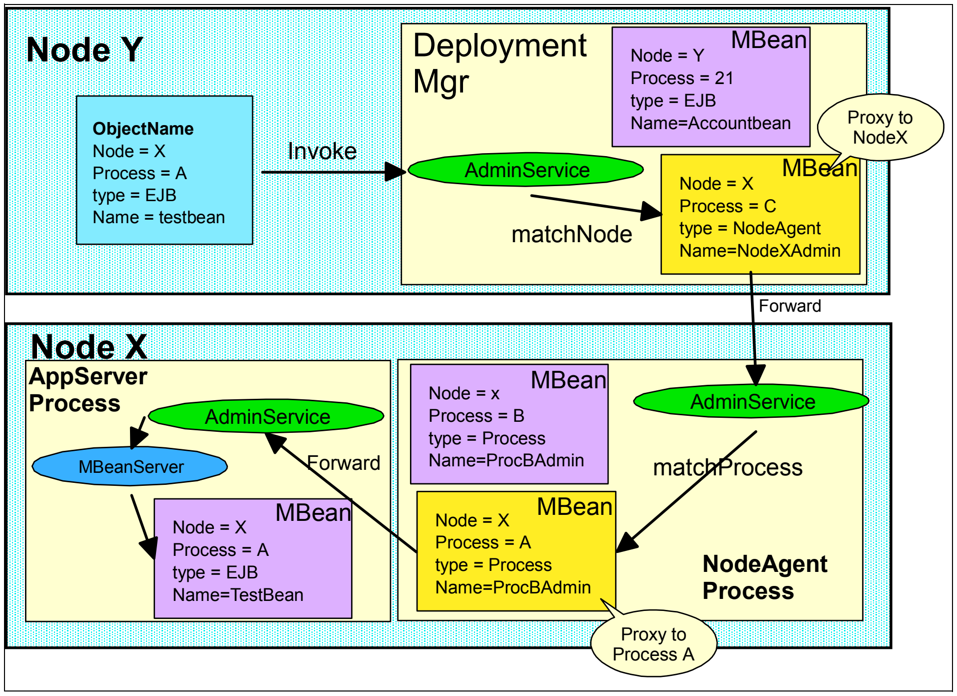


Figure 7-6 Distributed administration message routing

1. An object running on managed server (process) A of node Y sends the operation request to the Deployment Manager AdminService located on the same machine.
2. The Deployment Manager AdminService determines on which node the requested service is hosted (node X).
3. The Deployment Manager AdminService passes the request to the MBean acting as the proxy of the node X node agent.
4. The proxy MBean forwards the request to the AdminService of the node X node agent.

5. On node X, the node agent AdminService receives the request and determines managed server (process) the requested service is hosted on (process A).
6. The AdminService passes the request to the MBean acting as the proxy of the managed server.
7. The proxy MBean forwards the request to the AdminService of the managed server.
8. The managed server AdminService invokes the requested service via the local MBeanServer, which is responsible for all direct communication with MBeans hosted in that JVM.

For more information on JMX management beans (MBeans), see 16.2, “Java Management Extensions (JMX)” on page 780.

7.4.2 Administration points within a cell

Administration of configuration or application data can be performed at a number of points in the cell topology:

- Deployment Manager

  Manage everything under the cell. *This is the recommended approach*.

- Node agent

  Manages everything under the specific node. Other cell configuration cannot be managed from this point.

- Managed server

  Manages the server configuration, but not the node or cell configuration.

Note: The following issues should be kept in mind:

- The scripting administration client (wsadmin) can connect to the built-in AdminService of any Deployment Manager, node agent or managed server in a cell.
- The administrative console can only connect to a server that has the adminconsole.ear application installed. In a base installation, this is the default application server (server1). In Network Deployment, this is the Deployment Manager process. Node agents and managed servers in Network Deployment do not have the administrative console client application installed by default.
- Any changes made at the node or server level are only local and therefore will not be reflected in the cell's master repository. Local changes will be overridden when the next file synchronization (update) is performed for the node.
- Only changes made using the Deployment Manager (cell level) will be permanent for a node that is part of a cell.

7.4.3 Role-based administration

WebSphere V4 did not provide any access control granularity for its administrative subsystems. There was only one role, *Admin*, so that anyone with the administrative user name and password could perform all administrative functions.

In contrast, WebSphere Application Server V5 provides finer granularity of access control, through the provision of four administrative security roles:

- Monitor

  Can view the system state and configuration data, but cannot make any changes.
- Operator

  Has all the functions of Monitor as well as ability to make operational changes, for example start/stop servers.
- Configurator

  Has all the functions of Monitor as well as ability to make configurational changes.
- Administrator

  Has all the functions of Operator and Configurator.

7.4.4 Common administration functions

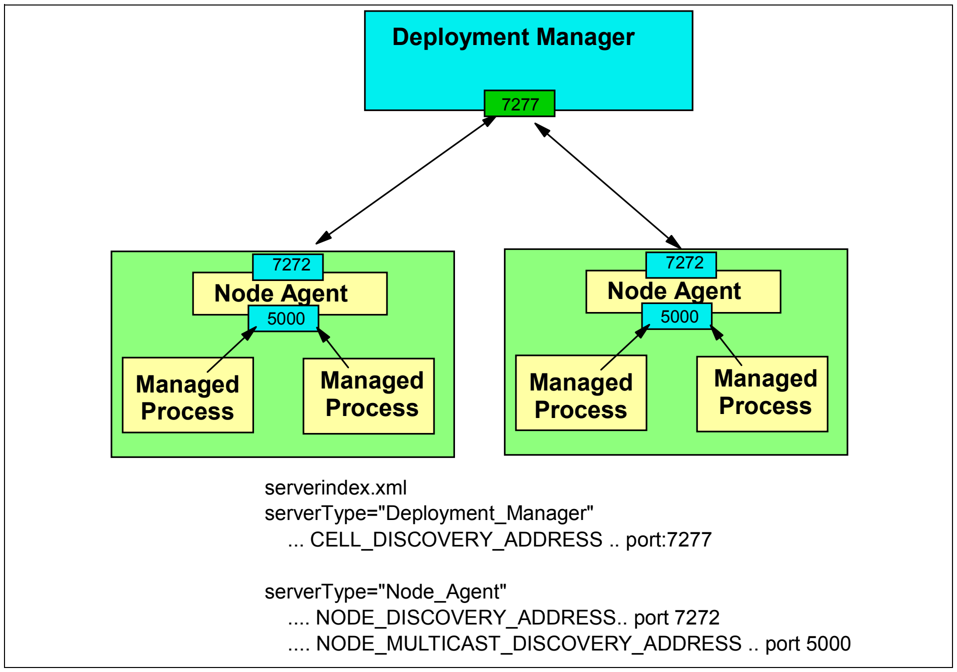
The distributed administration of Network Deployment is built upon the following common functions:

- ▶ Distributed process discovery
- ▶ Centralized changes to configuration data
- ▶ File synchronization
- ▶ Configuration file support
- ▶ Launching processes

Distributed process discovery

When a managed server begins its startup, it sends a discovery request message that allows other processes to discover its existence and establish communication channels with the process.

Figure 7-7 shows an example of the distributed discovery process for a WebSphere Application Server V5 topology containing two nodes that are located on different machines. If nodes are located on the same machine, they must be configured to use non-conflicting IP ports.



```
serverindex.xml
serverType="Deployment_Manager"
    ... CELL_DISCOVERY_ADDRESS .. port:7277

serverType="Node_Agent"
    .... NODE_DISCOVERY_ADDRESS.. port 7272
    .... NODE_MULTICAST_DISCOVERY_ADDRESS .. port 5000
```

Figure 7-7 Distributed discovery process

Key features

The key features of process discovery in the distributed WebSphere Application Server V5 environment include:

- Each node agent and Deployment Manager maintains status and configuration information by using discovery addresses (ports).
- On startup, WebSphere Application Server V5 processes (Deployment Manager, node agent and managed servers) discover other running components, and create communication channels between them, through the discovery addresses:
  - The master repository located on the Deployment Manager installation contains the serverindex.xml file for each node. The Deployment Manager reads this file on startup to determine the host name and IP port of each node agent's NODE_DISCOVERY_ADDRESS.
  - The copy of the configuration repository located on each node installation contains the serverindex.xml file for the Deployment Manager. The node agent reads this file on startup to determine the host name and IP port of the Deployment Manager's CELL_DISCOVERY_ADDRESS.
  - The copy of the configuration repository located on each node installation contains the serverindex.xml file for the node. Each managed server reads this file on startup to determine the host name and IP port of the node agent's NODE_MULTICAST_DISCOVERY_ADDRESS.
- Each server has its own copy of the configuration and application data necessary for startup of the runtime and the installed applications.

Rules for process startup

The order of process startup needs to adhere to the following rules:

- A node agent can be running while the Deployment Manager is not, and vice versa. When the stopped process is started, discovery will occur automatically.
- The Deployment Manager can be running while a managed server is not, and vice versa.

  Note: The execution of a managed server is not dependent on the presence of a running Deployment Manager. The Deployment Manager is only required for permanent configuration changes (changes written to the master repository).

- Unlike in WebSphere Application Server 5.0, in V5.1 an application server no longer fails at startup if the Node Agent has not been started first. The Node Agent is purely an administrative agent and is not involved in application serving functions.

Notes:

1. The node agent contains the Location Service Daemon (LSD) in which each application server registers itself on startup. If a node agent is restarted, then each application server on that node should also be restarted.
2. The JMS server process of a Network Deployment node does not run an Object Request Broker (ORB) service, and does not need to register on startup with the node agent LSD. As such, the JMS server does not need to be restarted if the node agent is restarted.

Cell discovery address

The cell discovery address defines the TCP/IP host name and port listened to by the Deployment Manager for discovery requests originating from node agents.

The cell discovery address is defined under the DEPLOYMENT_MANAGER stanza in the serverindex.xml file of the Deployment Manager node:

```
<WAS_ND_HOME>/config/cells/<cell>/nodes/<DM_hostname>Manager/serverindex.xml
```

An example is shown in Example 7-1.

Example 7-1 CELL_DISCOVERY_ADDRESS definition

```
<serverEntries xmi:id="ServerEntry_1" serverDisplayName="dmgr"
serverName="dmgr" serverType="DEPLOYMENT_MANAGER">
...
<specialEndpoints xmi:id="NamedEndPoint_1"
endPointName="CELL_DISCOVERY_ADDRESS">
       <endPoint xmi:id="EndPoint_1" host="mka0kkcf" port="7277"/>
</specialEndpoints>
...
</serverEntries>
```

Node discovery address

The node discovery address defines the TCP/IP host name and port listened to by a node agent for discovery requests originating from the Deployment Manager.

The node discovery address is defined under the NODE_AGENT stanza in the serverindex.xml file of the node:

```
<WAS_HOME>/config/cells/<cell>/nodes/<nodename>/serverindex.xml
```

An example is shown in Example 7-2 on page 220.

Example 7-2 NODE_DISCOVERY_ADDRESS definition

```
<serverEntries xmi:id="ServerEntry_2" serverDisplayName="Net1_JH"
serverName="Net1_JH" serverType="NODE_AGENT">
...
<specialEndpoints xmi:id="NamedEndPoint_1"
endPointName="NODE_DISCOVERY_ADDRESS">
        <endPoint xmi:id="EndPoint_1" host="mka0kkcf" port="7272"/>
</specialEndpoints>
...
</serverEntries>
```

Node multicast discovery address

The node multicast discovery address defines the multicast TCP/IP port listened to by the node agent for discovery requests originating from its managed servers (application servers and JMS server).

A multicast address is used to prevent the usage of a large number of IP ports for managed server to node agent discovery requests. Using multicast, a node agent can listen on a single IP port for any number of local servers.

The node multicast discovery address is defined under the NODE_AGENT stanza in the serverindex.xml file of the node:

```
<WAS_HOME>/config/cells/<cell>/nodes/<nodename>/serverindex.xml
```

An example is shown in Example 7-3.

Example 7-3 NODE_MULTICAST_DISCOVERY_ADDRESS definition

```
<serverEntries xmi:id="ServerEntry_2" serverDisplayName="Net1_JH"
serverName="Net1_JH" serverType="NODE_AGENT">
...
<specialEndpoints xmi:id="NamedEndPoint_1"
endPointName="NODE_MULTICAST_DISCOVERY_ADDRESS">
        <endPoint xmi:id="EndPoint_1" host="232.133.104.73" port="5000"/>
</specialEndpoints>
...
</serverEntries>
```

Important:

- The discovery service uses the InetAddress.getLocalHost() call to retrieve the IP address for the local machine's host name. The network configuration of each machine must be configured so that getLocalHost() does not return the loopback address (127.0.0.1). It must return the actual (real) IP address of the correctly chosen NIC.
- A multicast address is a logical address. Therefore it is not bound to an actual physical network interface, and will not be the same as the host name (or IP address) of the host on which the node agent is executed.
- Multicast host addresses must be within a special range (224.0.0.0 to 239.255.255.255) defined by the IP standards and must never be a host name value. The default for WebSphere node agents is 232.133.104.73.

Example discovery scenarios

1. Situation: The node agent is not running and the Deployment Manager starts:
   a. The Deployment Manager tries to determine if the node agent is running. No luck.
   b. When the node agent is started, it contacts the Deployment Manager, creates a communication channel, and synchronizes data.
2. Situation: The node agent starts but no managed servers are started:
   a. The node agent knows all about its managed servers and checks whether they are started. If so, it creates communication channels to these processes.
   b. When a managed server starts, it checks whether the node agent is started and then creates a communication channel to it.

Centralized changes to configuration and application data

IBM WebSphere Application Server Network Deployment provides a master repository of configuration and application data for the cell.

WebSphere Application Server V5 administrative clients are used to provide centralized functionality for:

- Modification of configuration settings in the master repository.
- Installation, update, and uninstallation of applications on application server(s) in the cell. In the process, the Enterprise Application Archive (EAR) files and deployment descriptors are also stored in the master repository.

Modifications must be saved to the repository before they are available for use by the WebSphere Application Server. Changes are committed to the master repository and then published out to the nodes of the cell. The files published to each of the nodes can include:

- All configuration and application data files of the cell.
- Only those configuration and application data files that apply to a particular node.
- Only those files documents that apply to a particular managed server on a particular node.

File synchronization

The file synchronization service is the administrative service responsible for keeping up to date the configuration and application data files that are distributed across the cell. The service runs in the Deployment Manager and node agents, and ensures that changes made to the master repository will be propagated out to the nodes, as necessary.

File synchronization occurs in the following cases:

- The node agent starts.
- The node agent periodically requests any changes from the Deployment Manager.
- The end user forces synchronization.
- The node agent's Auto Synchronization flag is set to `ON`.

During the synchronization operation, a node agent checks with the Deployment Manager to see if any files that apply to the node have been updated in the master repository. New or updated files are copied to the node, while any deleted files are also deleted from the node.

How files are identified for synchronization

When synchronization occurs, WebSphere must be able to identify the files that have changed and therefore, need to be synchronized. To do this WebSphere uses the following scheme:

- A calculated digest is kept by both the node agent and the Deployment Manager for each file in the configuration they manage . These digest values are stored in memory. If the digest for a file is recalculated and it does not match the digest stored in memory, this indicates the file has changed.
- An "epoch" for each folder in the repository and one for the overall repository is also stored in memory. These epochs are used to determine whether any files in the directory have changed. When a configuration file is altered through one of the WebSphere administration interfaces (administrative console, wsadmin, configuration service Java API, etc.) then the overall

repository epoch and the epoch for the folder in which that file resides is modified.

- During configuration synchronization operations, if the repository epoch has changed since the previous synchronize operation then individual folder epochs are compared. If the epochs for corresponding node and cell directories do not match, then the digests for all files in the directory are recalculated (including that changed file).

Synchronization scheduling

The scheduling of file synchronization is configured using an admin client. The available options are:

- Automatic synchronization

  Synchronization can be made to operate automatically by configuring the *File Synchronization Service* of the node agent. The alternatives are:

  – Periodically. The time interval can be set via the admin client.

  – When the node agent starts and the Deployment Manager is already running.

  – When the Deployment Manager starts and the node agent(s) is already running.

  Note: By default, automatic synchronization is *enabled*, with an interval of 60 seconds.

- Explicit/forced synchronization

  Synchronization can be explicitly forced at anytime via use of an administrative client.

- Startup synchronization

  If startup synchronization is enabled, by configuring the File Synchronization Service of the node agent, then a managed server's configuration files are synchronized as part of the managed server's startup.

Tip: In a production environment, the automatic synchronization interval should be increased so that processing and network overhead is reduced.

Synchronization process

The configuration and application data is synchronized out from the Deployment Manager to each of the node agents and in turn managed servers, in the cell, as depicted in Figure 7-8.

1. Admin client programs are used to modify the configuration and/or application settings. Whether or not the master repository is updated depends upon at which administration point the change was made. See 7.4.2, “Administration points within a cell” on page 215 for further details.
2. Individual nodes and servers synchronize the configuration and application data with the master repository.
3. Any changes made at the cell level are permanent.
4. Changes made at the node agent or managed server level are temporary. At the next data update time, the cell’s master files will be pushed out to the nodes and their managed servers.
5. The Deployment Manager checks-in/checks-out the configuration changes made to the master repository.

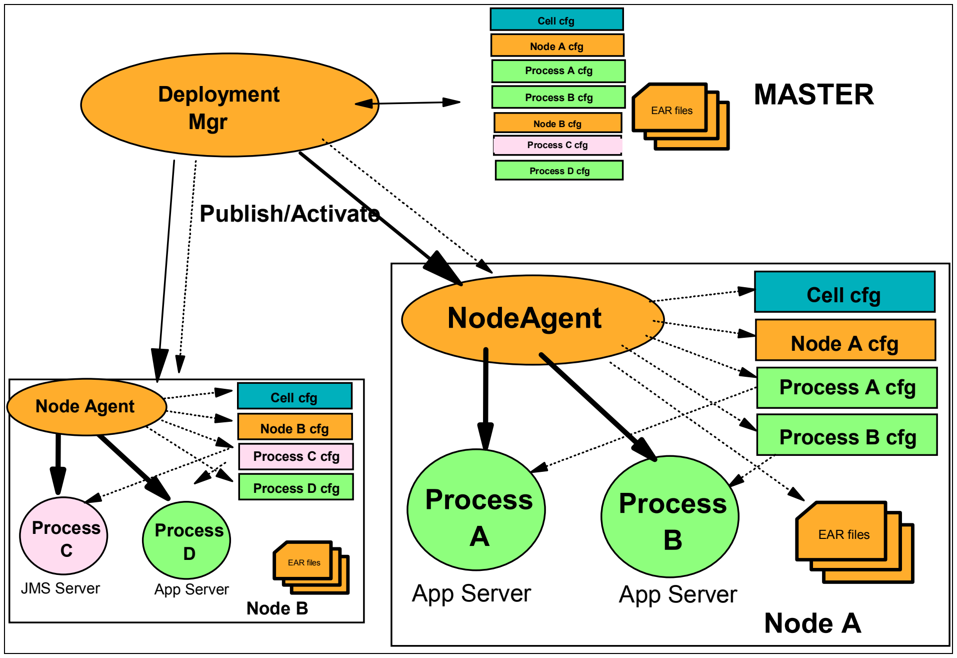


Figure 7-8 Configuration data synchronization

Each node will contain a separate instance (copy) of the repository. Each node's copy holds only the files required for that node, including:

- Cell and node-level configuration files necessary for node and managed server operation, for example the serverindex.xml file for each node in the cell.
- Application server configuration files for the application servers on that node.
- EAR files for the applications hosted by servers on that node.
- Application deployment descriptors for the applications hosted by servers on that node. These deployment descriptors contain the settings specified when the application was actually deployed.

Notes:

- The cell configuration document (cell.xml) is published to every node in the cell.
- Failures can occur that prevent publishing of changes from the master repository out to the appropriate node(s). For example, the node agent may not be running, in which case it is unable to provide the File Transfer Service required to support the publish operation.

  Node agents use a File Synchronization Service that checks for differences between pairs of configuration files (one in the node's local copy of the configuration, the other in the master repository) and provides a degree of automatic synchronization between the node and Deployment Manager based on a heuristic set of rules that decides which file contains the master configuration values.

Configuration file support

Each managed server has the data (configuration and application) necessary to start itself. The configuration data is in XML files and the application data is in EAR files.

WebSphere Application Server provides two administration clients that are to be used to edit the configuration of all components:

- The administrative console
- The wsadmin scripting client

Configuration changes made using one of these client programs will be checked by built-in validator code. Invalid configurations produce a warning so that corrective action can be taken.

Tip: Although the XML configuration files can be edited by hand, this method is strongly discouraged. Manual changes do not undergo the validation process until the server runtime attempts to run using the new configuration, for example on a restart. In this case, any config errors produce a warning message, and could result in the server runtime process failing to execute.

Launching processes

- The Deployment Manager process is launched using the `startManager` command.
- Node agent processes are launched using the `startNode` command.
- Managed server processes are launched using the `startServer` command.

Note: Each managed server has a configuration setting that determines whether it should be started automatically when the node agent is started. If this flag is enabled, then the node agent will also start these managed servers when it starts.

7.4.5 Required administration services

The distributed administration provides support for the services hosted by managed servers. MBean wrappers are used to expose the management services. The services hosted by each of the Network Deployment components is detailed in Table 7-1.

Table 7-1 Distributed administration services versus WebSphere process

Process	Services
Deployment Manager	AdminService Workload Management Cell Name Server
Node agent	AdminService File Transfer Service File Synchronization Service Node Name Server Location Service Daemon
Application server	AdminService Process Name Server Authentication Server
JMS server	AdminService Authentication Server

All processes host an AdminService (JMS agent) to support distributed administration. The JMX agent exposes the managed resources within the process for access and control from outside the process.

7.4.6 Required administrative applications

The operation of a Network Deployment environment requires that a number of applications be installed on the Deployment Manager in order to provide support for distributed administration.

The administrative console application (adminconsole.ear)

The administrative console is provided as a standard J2EE 1.3 compliant EAR file.

Note: By default, the node agent process is not configured to run a Web container. As such, the adminconsole application cannot be installed on a node agent process by default.

File Transfer Service (filetransfer.ear)

The Deployment Manager's File Transfer Service is invoked by node agents to transfer new and updated files from the master repository to a node's local copy of the configuration.

The service is provided as a standard J2EE 1.3 compliant EAR file.

7.5 Configuration and application data repository

The configuration and application data repository is a collection of files containing all the information necessary to configure and execute servers and their applications.

Most configuration files are in XML format, while application data is stored as EAR files and deployment descriptors.

7.5.1 Repository directory structure

The repository files are arranged in a set of cascading directories under <WAS_ND_HOME>/config, with each directory containing a number of files relating to different components of the WebSphere cell.

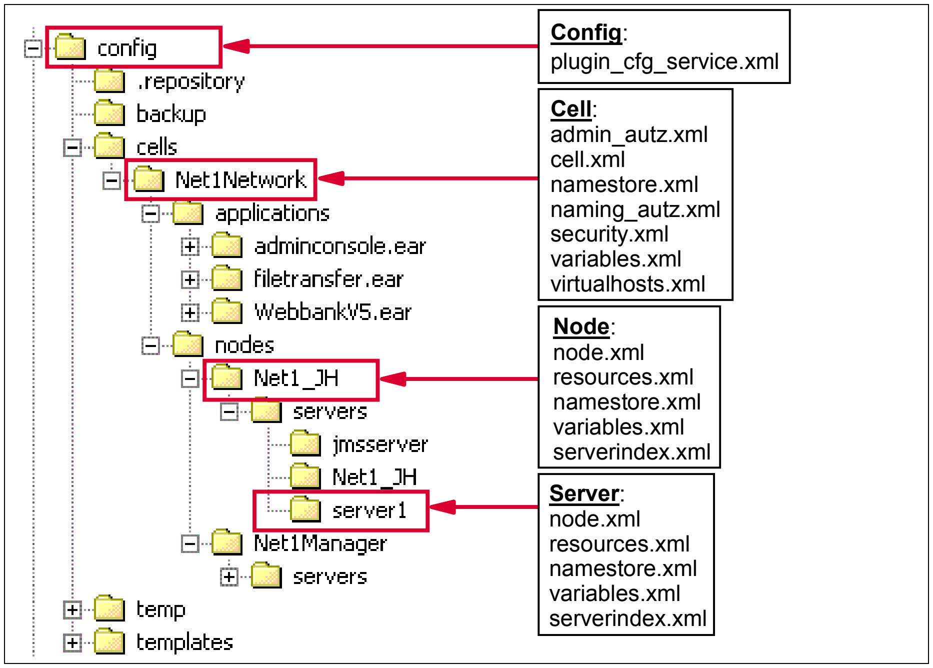


Figure 7-9 Repository directory structure

As shown in Figure 7-9, the <WAS_ND_HOME>/config directory is the root of the repository. It contains the following directory structure:

- cells/<cellname>/

  The root level of configuration for the cell. The directory contains a number of cell-level configuration settings files.

  Notes:

  1. The name of the cell in a Network Deployment installation is <*depmgr hostname*>*Network*. In the figure above, the Deployment Manager node name is *Net1*, so the cell directory is *Net1Network*.
  2. The name of the cell in a WAS-base installation is <*node name*>.

  There are three possible subdirectories under the cell directory. These subdirectories may not appear until after a cluster is created:

  – cells/<cellname>/applications/

    The applications directory contains one subdirectory for every application that has been deployed within the cell. The directory name in this case must match the deployed application name.

  - cells/<cellname>/nodes/

    The nodes directory contains the configuration settings for all nodes and servers managed as part of this cell. The directory contains one directory for each of the nodes managed by the cell.

  - cells/<cellname>/clusters/

    The clusters directory contains one directory for each of the clusters managed as part of this cell. Each cluster directory contains a single file, cluster.xml, which defines the application servers (of one or more nodes) that are members of the cluster.

- cells/<cellname>/nodes/<nodename>/

  Contains a servers directory as well as node-level configuration settings files.

- cells/<cellname>/nodes/<nodename>/servers/

  Contains one directory for each of the servers managed as part of this node.

- cells/<cellname>/nodes/<nodename>/servers/<servername>/

  Contains server-level configuration settings files.

7.5.2 Variable scoped files

Identically named files that exist at differing levels of the configuration hierarchy are termed "variable scoped" files. There are two uses for variable scoped files:

- Configuration data contained in a document at one level is logically combined with data from documents at other levels of the configuration hierarchy. In the case of conflicting definitions, the "most specific" value takes precedence. For example:

  If an identical entry exists in the files at the cell and node level (like a variable defined in both the cell and node's variables.xml), the entry at the node level takes precedence.

- Documents representing data that is not merged but is rather scoped to a specific level of the topology. For example:

  The namestore.xml document at the cell level contains the cell persistent portion of the name space, while the namestore.xml at the node level contains the node persistent root of the name space.

7.5.3 Manual editing of configuration files

Although configuration files can be edited manually, it is recommended that configuration changes be made using the administration tools whenever possible. However, at present these tools do not cover all possible configuration settings. As a result some configuration settings can only be changed through manual editing of configuration files.

See the following sections for an indication of which configuration files require manual editing.

Top-level configuration files

The top level of the configuration repository is contained in the following directory:

<WAS_ND_HOME>/config/

This directory typically contains the following configuration files.

Table 7-2 Top-level configuration files

File	Purpose	Variable scope (merged/not merged)	Manual editing required ? (Y/N)
plugin-cfg-service.xmi	Configuration of automatic plug-in regeneration settings.	not merged	N

The generated Web server plug-in is stored in the following location:

<WAS_ND_HOME>/config/cells

Table 7-3 Top-level configuration files

File	Purpose	Variable scope (merged/not merged)	Manual editing required ? (Y/N)
plugin-cfg.xml	Web server plug-in configuration settings.	not merged	N

Cell-level configuration files

The configuration of the cell is contained under the following directory:

<WAS_ND_HOME>/config/cells/<cellname>/

In this directory, there is a cell.xml file that contains configuration data specific to the cell. There are also several cell level files that contain configuration data for all managed servers running on that node. The cell-level directory typically contains the files listed in Table 7-4 on page 231.

Table 7-4 Cell-level configuration files

File	Purpose	Variable scope (merged/not merged)	Manual editing required ? (Y/N)
admin-authz.xml	Administrator access authorization settings	not merged	Y
cell.xml	Configuration data specific to the cell	not merged	N
filter.policy	Java 2 security policy file - permissions defined here will be filtered out in the node (app.policy) and application (was.policy) level files	not merged	Y
integral-jms-authorizations.xml	Manage access authorizations to resources owned by the Integral JMS provider	not merged	Y
multibroker.xml	Data replication service settings	not merged	N
namebindings.xml[1]	Name space bindings configured at this scope	merged	N
namestore.xml	Cell persistent portion of the name space	not merged	Y
naming-authz.xml	Naming authorization settings	not merged	Y
pmirm.xml	PMI request metrics	not merged	Y
resources.xml	Resources available for entire cell, for example JMS provider resources, JMS connection factories, JMS destinations, JDBC providers, etc.	merged	N
security.xml	Security configuration for the cell	not merged	N
variables.xml	Variable substitution values to use for processes running on the cell	merged	N
virtualhosts.xml	Virtual hosts and MIME types used by all servers in the cell	not merged	N

[1]The namebindings.xml file only exists at a particular level (cell, node, server) if one or more name space bindings have been configured at that level.

Cluster-level configuration files

If a Network Deployment cell has been configured to contain one (or more) application server clusters, then each cluster's configuration is contained in the directory:

<WAS_ND_HOME>/config/cells/<cellname>/clusters/<cluster name>/

This directory contains the following configuration file:

Table 7-5 Cluster-level configuration files

File	Purpose	Variable scope (merged/not merged)	Manual editing required? (Y/N)
cluster.xml	Identifies a cluster and its members (servers) and weights	N	N

Node-level configuration files

The configuration of each node is contained under the following directory:

<WAS_ND_HOME>/config/cells/<cellname>/nodes/<nodename>/

In each individual node directory, there is a node.xml file that contains configuration data specific to that node. There are also several node-level files that contain configuration data for all managed servers running on that node. The node-level directory typically contains the following files.

Table 7-6 Node-level configuration files

File	Purpose	Variable scope (merged/not merged)	Manual editing required ? (Y/N)
app.policy	Java 2 security file - access policies for all applications on this node	not merged	Y
library.policy	Java 2 security file - access policies for all libraries used by this node	not merged	Y
namebindings.xml[1]	Name space bindings configured at this scope	merged	N
[1]The namebindings.xml file only exists at a particular level (cell, node, server) if one or more name space bindings have been configured at that level.			

File	Purpose	Variable scope (merged/not merged)	Manual editing required ? (Y/N)
namestore.xml	Node persistent portion of the name space	not merged	Y
node.xml	Configuration data specific to the node	not merged	N
resources.xml	Resources available for entire node only	merged	N
serverindex.xml	Lists the servers associated with the node, the applications deployed to each server and the IP port numbers used by each server's services	not merged	N
spi.policy	Java 2 security file - access policies for all service provider interface (SPI) classes on this node	not merged	Y
variables.xml	Variable substitution values to use for processes running on the node	merged	N
[1]The namebindings.xml file only exists at a particular level (cell, node, server) if one or more name space bindings have been configured at that level.			

Server-level configuration files

The configuration of each managed server is found in the <WAS_ND_HOME>/config/cells/<cellname>/nodes/<nodename>/servers/<servername> directory.

In each individual server directory, there is a server.xml file that contains configuration data specific to that server. There are also several server level files that contain configuration data for the services hosted by the server. The server-level directory typically contains the following files.

Table 7-7 Server-level configuration files

File	Purpose	Variable scope (merged/not merged)
namebindings.xml[1]	Name space bindings configured at this scope	merged
namestore-cell.xml	Cell persistent portion of the name space	not merged
namestore-node.xml	Node persistent portion of the name space	not merged
resources.xml	Resources available on the server	merged
server.xml	Configuration data specific to the server	not merged
variables.xml	Variable substitution values to use for processes running on the server	merged
[1]The namebindings.xml file only exists at a particular level (cell, node, server) if one or more name space bindings have been configured at that level.		

Ensuring manual changes get synchronized

If a change to a configuration file is made by editing the file then the digest for the file is not recalculated because the epochs for the directories continue to match and the synchronization process will not recognize that the files have changed.

However, manual edits of configuration files in the master cell repository can be picked up if the repository is "reset" so that it re-reads all the files and recalculates all of the digests. You can reset either the master cell repository epoch or the node repository epoch.

- Resetting the master cell repository causes any manual changes made in the master configuration repository to be replicated to the nodes where the file is applicable.
- Resetting the node repository causes any manual changes to the local node files to be wiped out (overwritten) by whatever is in the master cell repository (regardless of whether the cell repository was changed or not) and any manual changes in the master repository will be picked up and brought down to the node.

The main difference between cell reset and node reset is that cell reset is likely to impact the entire cell, not just one node.

Note that this holds true for changes to installed applications as well. They are treated the same as other configuration files in the repository. For each installed application, there is an EAR file in the repository and also some configuration files associated with the deployment of the application.

If you manually change the EAR file (probably not the greatest idea unless you really know what you are doing) and reset the master cell repository, the changed EAR file will be replicated out to the nodes where it is configured to be served and will be expanded in the appropriate location on that node for the application server to find it. The application on that node will be stopped and restarted automatically so that whatever is changed is picked up and made available in the application server.

If you manually edit one of the deployment configuration files for the application (for instance, change something in the deployment.xml file for the app) and reset the repository, that change will be replicated to the applicable nodes and will be picked up the next time the application on that node is restarted.

Resetting the master cell repository

To performing a reset of the master cell repository do the following:

1. Open a command prompt and change to the <WAS_ND_HOME>/bin directory and start a wsadmin session

   ```
   cd c:\WebSphere\Deployment Manager\bin
   wsadmin
   ```

2. Enter the following:

   ```
   wsadmin>set config [$AdminControl queryNames
   *:*,type=ConfigRepository,process=dmgr]

   wsadmin>$AdminControl invoke $config refreshRepositoryEpoch
   ```

You will see a number returned by the refreshRepositoryEpoch operation (for instance: 1047961605195).

This resets the entire cell repository digest set. On the next synchronize operation, all files in the master cell repository will have their digests recalculated and any manual changes will be replicated to the applicable nodes.

Resetting the node repository

There are a multiple ways to reset a node repository for synchronization.

- The first method is using wsadmin. In a wsadmin session connected to the Deployment Manager (or node agent), enter the following:

  ```
  wsadmin>set config [$AdminControl queryNames
  *:*,type=ConfigRepository,process=nodeagent]

  wsadmin>$AdminControl invoke $config refreshRepositoryEpoch
  ```

  This resets the node digest set. Any file on the local node that does not match what is in the repository will be overwritten.

- The second way is to use the Network Deployment administrative console. Select **System Administration -> Nodes** to see a list of the nodes in the cell. You will notice Synchronize and Full Resynchronize buttons on the page. The Synchronize button causes a normal synchronize operation (no re-reading of the files). The Full Resynchronize button is the "reset and recalculate" function. Select the node or nodes to be updated with manual changes and click the **Full Resynchronize** button.
- Using the **`syncNode`** command. This command is a standalone program which runs separately from the node agent. It has no cache of epoch values which could be used for an "optimized" synchronize, therefore performing a complete synchronize. For this same reason, if you restart a node agent, the very first synchronize it performs will always be a complete synchronize. Note that this requires the node agent to be stopped.

  The **`syncNode`** command resides in the bin directory of the base install. To use the **`syncNode`** command, do the following:

  ```
  cd <WAS_HOME>\bin
  syncNode <cell_host>
  ```

Tip: The repository is flexible in that there is no predefined list of document types that it will permit. You can add any file you want into the repository. Perhaps you have some unique configuration data that needs to be used on all nodes. You could put it in the config/cells/<cell name> folder and it would be synchronized to all nodes. Or if it applies to just one node you could put it in the folder corresponding to that node and the synchronize will send it only to that node. And likewise for any additional documents in a server level folder.

For example, under normal circumstances, all application files are packaged in the EAR file for the application. However, consider a configuration file specific to an application. Any changes to that file would require that you update the EAR file and synchronize the entire application. One possibility is to put a properties file in the application deployment directory in the master configuration repository, so it is replicated to all nodes where the application is installed automatically (but the entire EAR isn't replicated). Then you could have an ExtensionMBean that updated the properties file in the master repository and normal synchronization would replicate just those changes out to the nodes without the need to synchronize the whole EAR (and restart the application).

7.5.4 Application data files

The master repository is also used to store the application binaries (EAR files) and deployment descriptors. This allows modified deployment descriptors to be kept in the repository, and allows system administrators to make application updates more automatic.

The root directory of the master repository (<WAS_ND_HOME>/config) contains the following directory structure used to hold application binaries and deployment settings:

- cells/<cellname>/applications/

  Contains a subdirectory for each application deployed in the cell. The names of the directories match the names of the deployed applications.

  Note: The name of the deployed application does not have to match the name of the original EAR file used to install it. Any name can be chosen when deploying a new application, as long as the name is unique across all applications in the cell.

- cells/<cellname>/applications/<appname>.ear

  Each application's directory in the master repository contains the following:

  – A copy of the original EAR, called <appname>.ear. This EAR does not contain any of the bindings specified during installation of the application.

  – A deployments directory, which contains a single <appname> directory used to contain the deployed application configuration.

- cells/<cellname>/applications/<appname>.ear/deployments/<appname>

  The deployment directory of each application contains the following:

  – deployment.xml

    Contains configuration data for the application deployment, including the allocation of application modules to application servers, and the module startup order.

  – META-INF/

    This directory contains the following:

    • application.xml

      J2EE standard application deployment descriptor.

    • ibm-application-bnd.xmi

      IBM WebSphere-specific application bindings.

    • ibm-application-ext.xmi

      IBM WebSphere-specific application extensions.

- was.policy

  Application-specific Java 2 security configuration. This file is optional. If not present, then the policy files defined at the node level will apply for the application.

Note: The deployment descriptors stored here in the repository contain the bindings chosen during application installation.

– Contains subdirectories for all application modules (WARs and EJB JARs), along with each module's deployment descriptors.

Note: The subdirectories for each module do not contain application binaries (JARs, classes, JSPs), only deployment descriptors and other configuration files.

7.5.5 Base versus Network Deployment repositories

The overall structure of the master repository is the same for both an IBM WebSphere Application Server stand-alone environment and an IBM WebSphere Application Server Network Deployment distributed environment.

The differences between the WebSphere Application Server base and Network Deployment master repositories are summarized in the following sections.

Base

- The master repository is held on a single machine. There are no copies of this specific repository on any other WebSphere Application Server V5 node.
- The repository contains a single cell and node.
- In a base installation, there is no node agent (since each application server is stand-alone) or JMS server (since each application server hosts the JMS server functionality). Therefore, the cells/<cellname>/nodes/ directory does not contain the JMS server (jmsserver) or node agent (nodeagent) server directories.
- Clusters are not supported in a base installation, and therefore will not contain the clusters directory or subdirectories.

Network Deployment

- The master repository is held on the node containing the Deployment Manager. It contains the master copies of the configuration and application data files for all nodes and servers in the cell.

- Each node also has a local copy of the configuration and application data files from the master repository that are relevant to the node.

  Changes can be made to the configuration files on a node, but the changes will be temporary. Such changes will be overwritten by the next file synchronization from the Deployment Manager. Permanent changes to the configuration require changes to the file(s) in the master repository.

  Configuration changes made to node repositories are not propagated up to the cell.
- The clusters directory will only be present if one (or more) application server clusters have actually been configured within the cell.
- In addition to the application server directories, there will exist the following server directories on each node:
  - jmsserver

    Contains the definition and settings for the JMS server process.
  - <nodename>

    Contains the definition and settings for the node agent process.
- The applications directory of the master repository contains the application data (binaries and deployment descriptors) for all applications deployed in the cell. The local copy of the applications directory on each node will only contain the directories and files for the applications actually deployed on application servers within that node.

7.5.6 Repository files used for application execution

The installation of an application onto a WebSphere Application Server V5 application server results in:

- The storage of the application binaries (EAR) and deployment descriptors within the master repository.
- The publishing of the application binaries and deployment descriptors to each node that will be hosting the application. These files are stored in the local copy of the repository on each node.

However, each node then installs applications ready for execution by exploding the EARs under the <WAS_HOME>/installedApps/<cellname>/ as follows:

- <WAS_HOME>/installedApps/<cellname>/

  This directory contains a subdirectory for each application deployed to the local node.

> **Note:** The name of each application's directory reflects the name under which the application is installed, not the name of the original EAR. For example, if an application is called *myapp*, then the *installedApps/<cellname>* directory will contain a *myapp.ear* subdirectory.

- <WAS_HOME>/installedApps/<cellname>/<appname>.ear/

  Each application specific directory contains the contents of the original EAR used to install the application.

  – The deployment descriptors from the original EAR. These descriptors do not contain any of the bindings specified during application deployment.

  – All application binaries (JARs, classes, JSPs).

Figure 7-10 summaries how the node's local copy of the repository contains the application's installed deployment descriptors, whereas the directory under installedApps contains the application binaries.

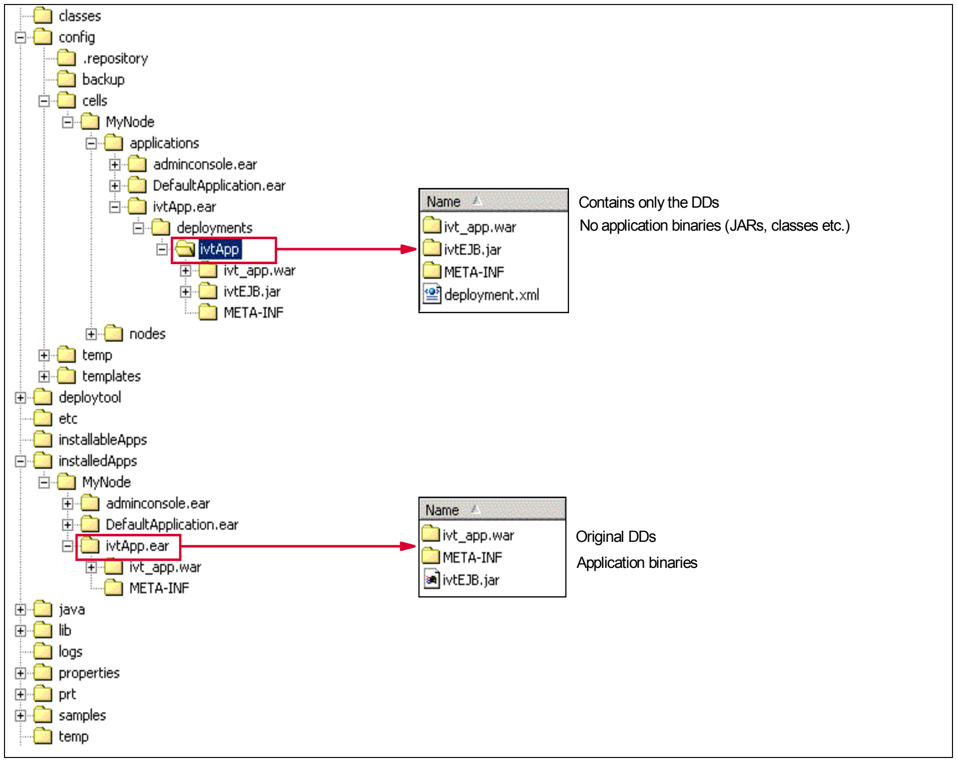


Figure 7-10 Location of application data files

By default, a WebSphere Application Server V5 application server executes an application by:

1. Loading the application binaries stored under:

   <WAS_HOME>/installedApps/<cellname>/<appname>.ear/

2. Configuring the application using the deployment descriptors stored under:

   <WAS_HOME>/config/cells/<cellname>/applications/<appname>.ear/deployments/<appname>

7.6 Application management

The system management of WebSphere Application Server V5 encompasses functionality relating to application management. The areas of application management addressed include:

- Application installation
- Application distribution
- Operational control
- Application upgrade and reinstall
- Application tools

For information on managing applications using the administrative console, see 8.6.6, “Managing enterprise applications” on page 314. For managing applications with wsadmin, see 16.4.6, “Managing enterprise applications” on page 806.

7.6.1 Application installation

WebSphere Application Server V5 provides a common installation architecture across all editions, as compared to WebSphere V4 where different tools and techniques had to be used depending upon the edition.

The steps performed for application installation involve:

- Validation of the application archive.
- Collection of configuration information from the end user, either interactively (via wsadmin or the administrative console) or non-interactively (via a prepared script).
- Registration of the application with the cell.
- Distribution of configuration changes and application binaries to each of the nodes controlling application servers chosen to host the application’s modules.

For information on application installation, see Chapter 14, “Deploying applications” on page 647.

7.6.2 Application distribution

IBM WebSphere Application Server provides automatic distribution of applications, which involves the transfer of application configuration and binaries to servers that the application runs on in a multi-node environment.

This is a much-improved feature in comparison to WebSphere 4.0, where the administrator was expected to copy and expand the application binaries on all the nodes where the application is installed, before the application could be run.

Application management performs distribution in the following cases:

- Application installation
- Application reinstall or upgrade
- Addition of a new member server to a server cluster

  When an application server is added as a member to a server cluster, the modules installed on other members are also installed on the new member.
- Rebinding of a module to a server or a server cluster

  After an application is installed in the WebSphere cell, its individual modules can be rebound to another stand-alone server or server cluster. When a module is bound to a server and if the node that the server runs on does not already have the application binaries, then application transfer logic is executed.

7.6.3 Application operational control

Applications and modules are represented using JMX MBeans in the system management architecture. For an introduction to JMX, see 16.2, “Java Management Extensions (JMX)” on page 780.

The following application management is possible using the WebSphere administrative tools:

- Starting and stopping applications.
- Viewing and editing of an application’s configuration

7.6.4 Application upgrade

IBM WebSphere Application Server provides built-in functionality to support the upgrade of currently installed enterprise applications. Two different upgrade approaches are provided, each suitable in different circumstances.

Full application upgrade

This approach involves using the WebSphere administrative tools to replace a currently installed enterprise application with a new version of the application's Enterprise Application Archive (EAR).

The major features of this application upgrade approach are:

1. Application updates are packaged as a new Enterprise Application Archive (EAR). Typical application changes include:
   - Editing J2EE deployment descriptors.
   - Adding or importing J2EE modules (Web, EJB, client) into an application.
   - Adding EJBs, Web components (images, HTML or JSP).
   - Adding or updating compiled Java code (JARs and class files).
   - Adding or editing property files.
2. The WebSphere administative tool (Web admin console or wsadmin) is used to select the currently installed application, then to choose the application upgrade command.
3. The administrative tool provides the same installation steps as during the installation of a new application. However, in the upgrade case the displayed bindings are a merger of the bindings contained in the new version (from the EAR) and the currently installed version. The meger process can be summarized as follows:
   - If the new version has bindings for application artifacts, then these will be part of the merged binding information.
   - If new bindings are not present for existing application artifacts, then installed version's bindings will be used.
   - If bindings are not present in the old version and if the default binding generation option is enabled, then the default bindings will be part of the merged binding information.
4. The user has the opportunity to change any of the merged deployment bindings before finally applying the upgrade. For example, the user may choose to change which application servers or clusters the upgraded application will be deployed upon.
5. The application binaries and bindings of the old version will be deleted from the master repository, then the application binaries and bindings of the new version will be copied to the master repository.
6. The configuration and application binaries on each node will be updated with the necessary changes when the Deployment Manager next synchronizes each node.

Important:

- On each node, the application server will be stopped immediately before the application binaries and configuration are updated during node synchronization. This means that the application instance running on a node will be unavailable during that node's synchronization operation.
- On completion of a node's synchronization, the application server on that node will be restarted.
- Since nodes are synchronized one at a time, until all affected nodes have been re-synchronized, some nodes will be running the new version of the application and some will be running the old version.

Hot deployment and dynamic reloading

This approach involves making changes to an application and its contents without having to restart the application server.

The types of changes can be categorized as follows:

- **Hot deployment of new components**

  Process of adding new components (such as WAR files, EJB JAR files, EJBs, servlets, and JSP files) to a running application server without having to stop and then restart the application server.

  However, in most cases such changes will require the application itself to be restarted, so that the application server runtime reloads the application and its changes.

- **Dynamic reloading of existing components**

  Ability to change an existing component without the need to restart the application server in order for the change to take effect. Dynamic reloading can involve:

  – Changes to the implementation of an application component, such as changing the implementation of a servlet.

  – Changes to the settings of the application, such as changing the deployment descriptor for a Web module.

In this upgrade approach, the application is generally not updated using the WebSphere administrative tools. Instead, the application files (binaries and/or configuration) are manually added, edited, or deleted in the master repository on the file system.

Note: Although the application files can be manually edited on one (or more) of the nodes, these changes will be overwritten the next time the node synchronizes its configuration with the Deployment Manager. Therefore, it is recommended that manual editing of an application's files should only be performed in the master repository, located on the Deployment Manager machine.

See the "Updating applications" section of the InfoCenter for a detailed list of the application changes that can be made using the hot deployment and dynamic reloading approaches.

7.6.5 Application assembly tools

IBM WebSphere Application Server V5.1 provides the Assembly Toolkit for assembling enterprise application modules, replacing the Application Assembly Tool (AAT) provided with prior versions.

For detailed information on assembling and packaging applications using this tool, see Chapter 13, "WebSphere specific packaging options" on page 617.

7.7 Common system management tasks

This section summarizes some of the most common system management tasks:

- Adding a node to a cell.
- Removing a node from a cell.
- Force synchronization of a node's configuration.
- Cleaning up a node.
- Backing up a node configuration.
- Restoring a node configuration.
- Backing up the cell configuration.
- Restoring the cell configuration.
- Creating multiple instances (nodes) on a single machine.
- Starting a Network Deployment environment.
- Stopping a Network Deployment environment.
- Enabling process restart on failure.
- Installing an application.
- Updating an application.

7.7.1 Adding a node to a cell

Use the **addNode** command to add a stand-alone node to an existing cell. The principles of this operation are shown in Figure 7-11. See A.5, "addNode" on page 843 for detailed information on using the **addNode** command.

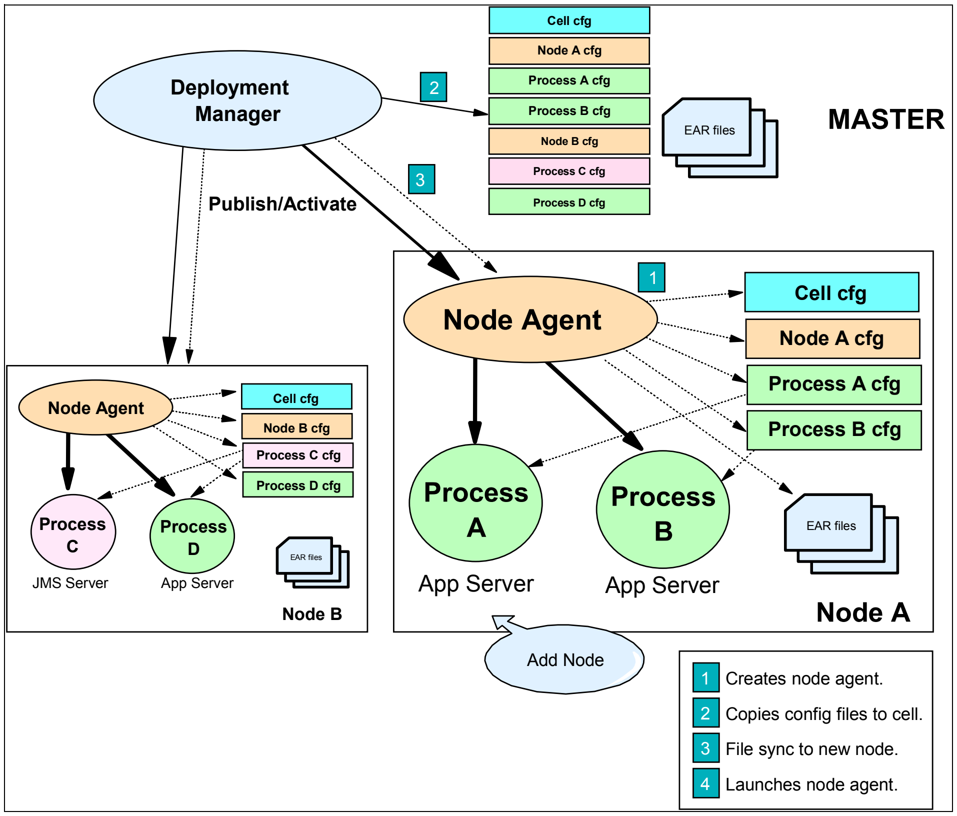


Figure 7-11 Add node to cell

7.7.2 Removing a node from a cell

Use the **removeNode** command to detach a node from a cell and return it to a stand-alone configuration. See A.6, "removeNode" on page 847 for detailed information on using the **removeNode** command.

7.7.3 Forcing synchronization of the node configuration

An administrator may be required to explicitly force a resynchronization of a node's configuration, and thereby update its local configuration files to reflect the current contents of the master repository, for example if the node agent was not running when changes were made to the cell repository.

In such instances, the `syncNode` command is used to force the resynchronization of a node with its cell. See A.7, “syncNode” on page 850 for detailed information on using the `syncNode` command.

7.7.4 Cleaning up a node

WebSphere Application Server V5 does not provide a command specifically for cleaning up an installation, for example removing log files. As such, an administrator must use a manual process.

Suggested directories that can be cleaned up include:

1. On the Deployment Manager machine:
   – <WAS_ND_HOME>/wstemp

     Contains the temporary working areas for unsaved (uncommitted) workspace changes for each administrator.

   – <WAS_ND_HOME>/logs/dmgr/

     Contains the log files generated by the Deployment Manager process.

   – <WAS_ND_HOME>/logs/ffdc/

     Contains logs generated by the First Failure Data Collection (FFDC) subsystem.

2. On each node machine:
   – <WAS_HOME>/logs/<server name>/

     Contains the log files generated by this server process.

   – <WAS_HOME>/logs/nodeagent/

     Contains the log files generated by the node agent process.

   – <WAS_HOME>/logs/ffdc/

     Contains logs generated by the First Failure Data Collection (FFDC) subsystem.

Important: Stop all managed servers on a machine before deleting any logs on that machine.

7.7.5 Backing up a node configuration

Use the **`backupConfig`** command to back up the configuration of a node. The command backs up an entire copy of the node's config directory to a ZIP archive.

See A.14, "backupConfig" on page 865 for detailed information on using the **`backupConfig`** command.

7.7.6 Restoring a node configuration

Use the **`restoreConfig`** command to restore the configuration of a node using an archive previously generated using **`backupConfig`**. The command restores the entire contents under the node's config directory.

See A.15, "restoreConfig" on page 868 for detailed information on using the **`restoreConfig`** command.

7.7.7 Backing up the cell configuration

The master cell configuration repository can be backed up by running the **`backupConfig`** command on the machine hosting the Deployment Manager. The command backs up an entire copy of the node's config directory to a ZIP archive.

See A.14, "backupConfig" on page 865 for detailed information on using the **`backupConfig`** command.

7.7.8 Restoring the cell configuration

The master cell configuration repository can be restored by running **`restoreConfig`** on the machine hosting the Deployment Manager. This requires a backup archive previously generated by **`backupConfig`** on the same machine.

The command restores the entire contents under the Deployment Manager's config directory.

See A.15, "restoreConfig" on page 868 for detailed information on using the **`restoreConfig`** command.

7.7.9 Creating multiple instances (nodes) on a single machine

WebSphere Application Server V5 provides the **`wsinstance`** command as a means by which multiple instances (nodes) can be set up and run on a single machine using a single IBM WebSphere Application Server installation.

Each instance uses a separate configuration but uses a single installation of the WebSphere binaries. Each instance is distinguished by its base path, its own directory structure, and its own setupCmdLine script to configure its command-line environment.

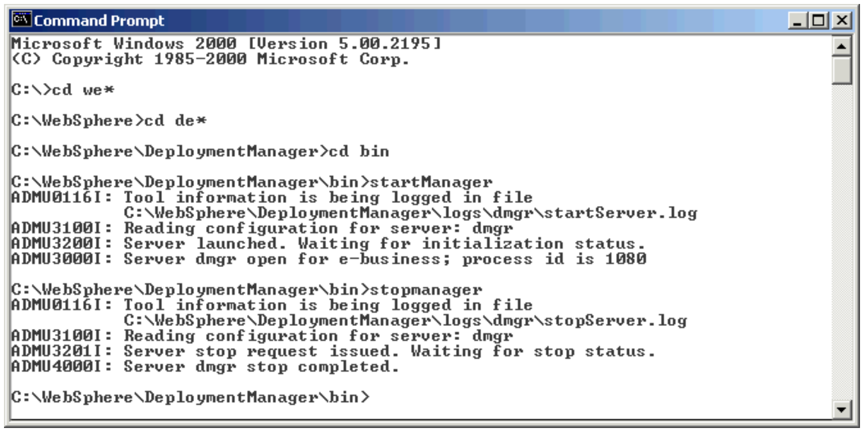
See A.16, “wsinstance” on page 870 for detailed information on using the **wsinstance** command.

7.7.10 Starting the Network Deployment environment

The recommended procedure for starting a Network Deployment environment involves the following steps:

1. On the Deployment Manager machine:
   a. Change directory to the bin directory of the Network Deployment installation.
   b. Use the **startManager** command to start the Deployment Manager (dmgr) process. See A.1, “startManager” on page 834, for command-line options.

   If successful, you will see the process ID for the network Deployment Manager process displayed on the window. See Figure 7-4 on page 213.



```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>cd we*

C:\WebSphere>cd de*

C:\WebSphere\DeploymentManager>cd bin

C:\WebSphere\DeploymentManager\bin>startManager
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\DeploymentManager\logs\dmgr\startServer.log
ADMU3100I: Reading configuration for server: dmgr
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server dmgr open for e-business; process id is 1080

C:\WebSphere\DeploymentManager\bin>stopmanager
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\DeploymentManager\logs\dmgr\stopServer.log
ADMU3100I: Reading configuration for server: dmgr
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server dmgr stop completed.

C:\WebSphere\DeploymentManager\bin>
```

Figure 7-12 Starting and stopping the Deployment Manager from the command line

   If there are any errors, check the log file for the dmgr process:

   ```
   <WAS_ND_HOME>/logs/dmgr/SystemOut.log
   ```

2. On each node:
   a. Change directory to the bin directory of the base application server installation for that node.

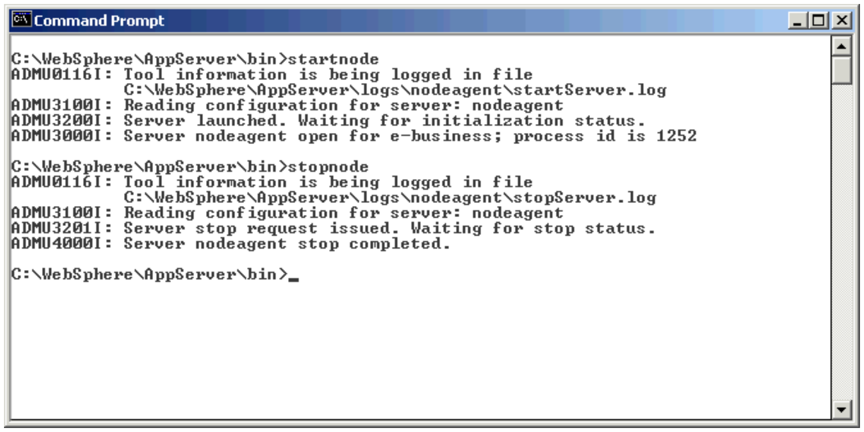
Note: Installation of multiple nodes in a single physical machine requires the use of the **wsinstance** command. See 7.7.9, “Creating multiple instances (nodes) on a single machine” on page 248 for details.

b. Run the **startNode** command.

If successful, the node agent server process ID will be displayed on the window, as shown in Figure 7-13.

If there are any errors check the log file for the node agent process:

```
<WAS_HOME>/logs/nodeagent/SystemOut.log
```



```
C:\WebSphere\AppServer\bin>startnode
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\nodeagent\startServer.log
ADMU3100I: Reading configuration for server: nodeagent
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server nodeagent open for e-business; process id is 1252

C:\WebSphere\AppServer\bin>stopnode
ADMU0116I: Tool information is being logged in file
           C:\WebSphere\AppServer\logs\nodeagent\stopServer.log
ADMU3100I: Reading configuration for server: nodeagent
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server nodeagent stop completed.

C:\WebSphere\AppServer\bin>_
```

Figure 7-13 Starting and stopping the node agent from the command line

c. Use the **startServer** command to start each of the application server processes on the node.

d. Use the **startServer** command to start the jmsserver process on the node.

e. Check the node status by running the **serverStatus -all** command.

Note: Managed servers can be configured to be automatically started on the start of the node agent. If already running, each application server must be restarted anyway as each must register on startup with the Location Service Daemon (LSD) of the node agent.

3. Repeat (2) for each and every node associated with this Deployment Manager.

7.7.11 Stopping the Network Deployment environment

The sequence of steps to follow to correctly stop a Network Deployment environment involves the following steps:

1. On each node agent machine:
   a. Use the **`stopServer`** command to stop each of the application server processes on the node.
   b. Use the **`stopServer`** to stop the jmsserver process on the node.
   c. Use the **`stopNode`** command to stop the node agent process.
      i. Change directory to the bin directory of the base application server installation for that node.
      ii. Run the **`stopNode`** command.

      If successful, the message "`Server <node_agent> stop completed`" will be output to the console, as shown in Figure 7-13 on page 250.

      If there are any errors check the log file for the node agent process:

      <WAS_ND_HOME>/logs/dmgr/SystemOut.log

      Important: Stopping the node agent doesn't stop the application servers managed by that node. However, because the node agent runs the Location Service Daemon, it is recommended that you restart the application servers after restarting the node agent.

   d. Check the node status by running the **`serverStatus -all`** command.
2. Repeat (2) for each and every node associated with this Deployment Manager.
3. On the Deployment Manager machine:
   a. Change directory to the bin directory of the Network Deployment installation.
   b. Use the **`stopManager`** command to stop the Deployment Manager (dmgr) process.

   The message "`Server dmgr stop completed`" will be seen if the server stopped successfully, as shown in Figure 7-12 on page 249.

   If there are any errors, check the log file for the dmgr process:

   <WAS_ND_HOME>/logs/dmgr/SystemOut.log

Note: Stopping the Deployment Manager does not stop any node agents.

7.7.12 Enabling process restart on failure

Unlike IBM WebSphere Application Server V4, the V5 software does not have either:

- A nanny process to monitor whether the AdminServer process is running, and restart it if the process has failed.
- An AdminServer process to monitor whether each application server process is running, and restart it if the process has failed.

Instead, IBM WebSphere Application Server V5 uses the native operating system functionality to restart a failed process:

Windows 2000

The administrator can choose to register one (or more) of the IBM WebSphere Application Server V5 processes on a machine as a Windows Service. Windows will then automatically attempt to restart the service if it fails.

Syntax

The syntax is as follows:

```
WASService.exe (with no arguments starts the service)
             || -add <service name>
                -serverName <Server>
                    [-wasHome <Websphere Install Directory>]
                    [-configRoot <Config Repository Directory>]
                    [-startArgs <additional start arguments>]
                    [-stopArgs <additional stop arguments>]
                    [-userid <execution id> -password <password>]
                    [-logFile <service log file>]
                    [-logRoot <server's log directory>]
                    [-userScript <setupCmdLine script file>]
                    [-restart <true | false>]
             || -remove <service name>
             || -start <service name>
             || -stop <service name>
             || -status <service name>
```

Notes

1. When adding a new service, only the `serverName` argument is mandatory.
2. Use unique names for the WebSphere server processes registered as Windows services. The services will be listed in the Windows Services control panel as:

   IBM WebSphere Application Server V5 - <service name>

3. To configure the node agent as a Windows service, use a serverName that is the same as the nodeName. That is the convention for node agent server process names.
4. To configure the Deployment Manager as a Windows service, use the Deployment Manager specific nodeName.

Examples

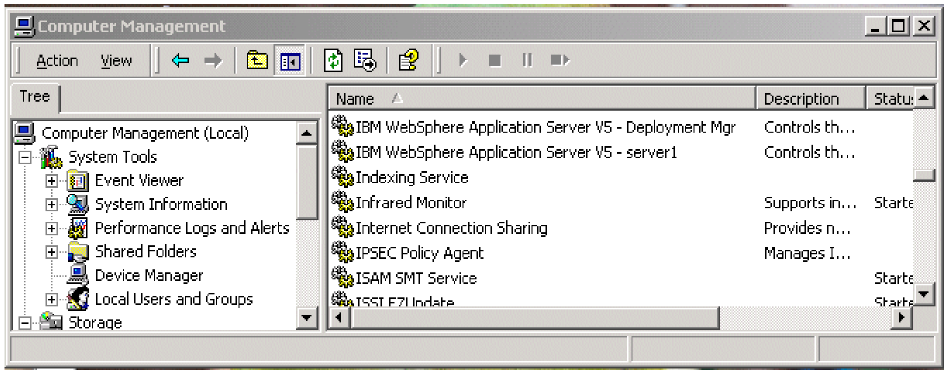
Example 7-4 shows the use of the WASService command to add the Deployment Manager as a Windows service.

Example 7-4 Registering a Deployment Manager as a Windows service

```
C:\WebSphere\DeploymentManager\bin>wasservice -add "Deployment Mgr" -serverName
dmgr -wasHome "c:\websphere\deploymentmanager" -restart true

Adding Service: Deployment Mgr
        Config Root: c:\websphere\deploymentmanager\config
        Server Name: dmgr
        Was Home: c:\websphere\deploymentmanager\
        Start Args:
        Restart: 1
IBM WebSphere Application Server V5 - Deployment Mgr service successfully
added.
```

Note that the service name added (Figure 7-14) will be “IBM WebSphere Application Server V5 - ” concatenated with the name you specified for the service name.



Name	Description	Status
IBM WebSphere Application Server V5 - Deployment Mgr	Controls th...	
IBM WebSphere Application Server V5 - server1	Controls th...	
Indexing Service		
Infrared Monitor	Supports in...	Starte
Internet Connection Sharing	Provides n...	
IPSEC Policy Agent	Manages I...	
ISAM SMT Service		Starte
ISSI EZUpdate		Starte

Figure 7-14 New service

If you remove the service using the WASService -remove command, specify only the latter portion of the name (Example 7-5).

Example 7-5 Removing a service

```
C:\WebSphere\DeploymentManager\bin>wasservice -remove "Deployment Mgr"

Remove Service: Deployment Mgr
Successfully removed
service
```

The commands shown in Example 7-6 are used on a WebSphere Application Server node to add the node agent and a server as Windows services.

Example 7-6 Registering WebSphere processes as Windows services

```
C:\> cd \ibm\was\AppServer\bin

C:\WebSphere\AppServer\bin>wasservice -add NodeAgent -serverName nodeagent
-wasHome "c:\WebSphere\AppServer" -restart true
Adding Service: NodeAgent
        Config Root: c:\WebSphere\AppServer\config
        Server Name: nodeagent
        Was Home: c:\WebSphere\AppServer\
        Start Args:
        Restart: 1
IBM WebSphere Application Server V5 - NodeAgent service successfully added.

C:\WebSphere\AppServer\bin>wasservice -add Server2 -serverName server2 -wasHome
"c:\WebSphere\
AppServer" -restart true
Adding Service: Server2
        Config Root: c:\WebSphere\AppServer\config
        Server Name: server2
        Was Home: c:\WebSphere\AppServer\
        Start Args:
        Restart: 1
IBM WebSphere Application Server V5 - Server2 service successfully added.
```

UNIX/Linux

The administrator can choose to include entries in inittab for one (or more) of the WebSphere Application Server V5 processes on a machine, as shown in Example 7-7. Each such process will then be automatically restarted if it has failed.

Example 7-7 Inittab contents for process restart

```
On Deployment Manager machine:
   ws1:23:respawn:/usr/WebSphere/DeploymentManager/bin/startManager.sh

On node machine:
```

```
ws1:23:respawn:/usr/WebSphere/AppServer/bin/startNode.sh
ws2:23:respawn:/usr/WebSphere/AppServer/bin/startServer.sh nodename_server1
ws3:23:respawn:/usr/WebSphere/AppServer/bin/startServer.sh nodename_server2
ws4:23:respawn:/usr/WebSphere/AppServer/bin/startServer.sh nodename_server2
```

8

8. WebSphere administration basics

In this chapter, we introduce the WebSphere administrative console and describe some of the basic tasks that are commonly performed by WebSphere administrators. The tasks we look at include:

- Using the administrative console.
- Working with cells, nodes, applications servers and enterprise applications.
- Viewing installed enterprise application properties, including servlet URLs.

8.1 Introducing the WebSphere administrative console

The WebSphere administrative console is the graphical, Web-based tool that you use to configure and manage an entire WebSphere cell. It supports the full range of product administrative activities, such as creating and managing resources, applications, viewing product messages, etc. The administrative console is a standard J2EE 1.3 Web application running under the Deployment Manager server, *dmgr,* and is installed by default when the Network Deployment Manager is installed.

Note: The administrative console application also gets installed when you install a base instance of the IBM WebSphere Application Server on a node. However, as the node is added to a Network Deployment cell, the administrative console application is removed from the node.

The administrative console provides centralized administration of multiple nodes, and allows nodes on multiple machines to be administered. The configuration data for a Network Deployment cell is a set of XML documents arranged in a set of cascading directories under <WAS_ND_HOME>/config directory. With the administrative console, we load and make changes to the master repository XML configuration files. It is the Deployment Manager's responsibility to push those changes to the local XML repositories on the nodes.

Note: In the Network Deployment environment, it is possible to install the administrative console on any of the nodes of the cell and it to the server. This allows for local administration of the server. However, any changes made to the server configuration will be temporary. At the next scheduled data update time (file synchronization time), the Deployment Manager pushes the master configuration data to the nodes and any changes made at the server level are lost. For changes to be permanent, they must be performed at the Deployment Manager level.

In order for the administrative console to run, the dmgr server must be running in the node where Network Deployment (and therefore the administrative console) is installed.

In order for the changes to the master repository to be pushed to the nodes, the node agents must also be running in the nodes where the WebSphere Application Server V5 instances are installed.

Note: WebSphere scripting can also be used to configure and modify configuration settings.

In WebSphere Application Server V5, the administrative console groups administrative tasks as follows:

- Servers
- Applications
- Resources
- Security
- Environment
- System administration
- Troubleshooting

Note: Users new to J2EE should be aware that there have been major changes to the WebSphere administrative console between WebSphere V3.5 and WebSphere V5. Familiarity with the concepts underlying a J2EE runtime environment is required in order to effectively manage a WebSphere V5 environment.

8.2 Starting the administrative console

In Network Deployment, the administrative console is deployed as a J2EE application:

- Application binaries

  <WAS_ND_HOME>/installedApps/<CELL>/adminconsole.ear

- Application configuration:

  <WAS_ND_HOME>/config/cells/<CELL>/applications/adminconsole.ear

The application is managed by the Deployment Manager process, dmgr.

To start the administrative console:

1. Make sure that Deployment Manager, dmgr, is running:
   - Windows: **`<WAS_ND_HOME>\bin\serverStatus -all`**
   - UNIX: **`<WAS_ND_HOME>/bin/serverStatus.sh -all`**
2. If the dmgr status is not “STARTED”, start it with the following command:
   - On Windows: **`<WAS_ND_HOME>\bin\startManager`**
   - On UNIX: **`<WAS_ND_HOME>/bin/startManager.sh`**

   Note: In this section, we assume that a connection is made to the administrative console installed in the Network Deployment node.

3. Open a Web browser to the URL of the administrative console. The default is port is 9090 for HTTP and 9043 for HTTPS.

   – `http://<DM_hostname>:9090/admin`
   – `https://<DM_hostname>:9043/admin`

   where `<DM_hostname>` is the host name for the machine running the Deployment Manager process, dmgr.

   Note: If you need two concurrent sessions on the same client machine, access the administrative console from two different browser types, whether or not you use the same user ID. This will allow for two different HTTP session objects.

4. The administrative console will load into the browser and you will be asked to log in.

8.2.1 Logging into the administrative console

The user ID specified during login is used to track configuration changes made by the user. This allows you to recover from unsaved session changes made under the same user ID, for example when a session times out or the user closes the Web browser without saving.

The user ID used for login depends on whether WebSphere global security is enabled.

- **No security:** If global security is not enabled, you can enter any user ID, valid or not to log in to the administrative console. The user ID is used to track changes to the configuration but is not authenticated.
- **WebSphere global security is enabled**: If global security is enabled, you must enter a valid user ID and password.

A user ID must be unique to the Deployment Manager. If you enter an ID that is already in use (and in session), you will receive the message `Another user is currently logged with the same User ID` and you will be prompted to do one of the following:

- Force the existing user ID out of session. You will be allowed to recover changes that were made in the other user’s session.
- Wait for the existing user ID to log out or time out of the session.
- Specify a different user ID.

Note: This message will appear if a previous session ended without a logout, for example if the user closed a Web browser during a session and did not logout first or if the session timed out.

Recovering from an interrupted session

Until you save the configuration changes you make during a session, the changes do not become effective. If a session is closed without a save being done for the configuration changes made during the session, these changes are remembered and you are given the chance to pick up where you left off.

When unsaved changes for the user ID exist during login, you will be prompted to do one of the following:

- Work with the master configuration

  When enabled, specifies that you want to use the last saved administrative configuration. Changes made to the user's session since the last saving of the administrative configuration will be lost.

- Recover changes made in prior session

  When enabled, specifies that you want to use the same administrative configuration last used for the user's session. Recovers all changes made by the user since the last saving of the administrative configuration for the user's session.

Tip: You may want to change the session timeout for the administrative console application. This is the time for the session to time out when the console is not used. The default is 30 minutes. To change the session timeout value:

- Expand **Applications** and select **Enterprise Applications**.
- Click the **adminconsole** application link.
- Click **Session Management** under the Configuration tab.
- Find **Session Timeout** and change the minutes.
- Click **OK**.

As you work with the configuration, the original configuration file and the new configuration file are stored in a user workspace at:

```
<WAS_ND_HOME>/wstemp/<user>/workspace/cells/<cell>
```

Once you have saved the changes, they are removed from the workspace.

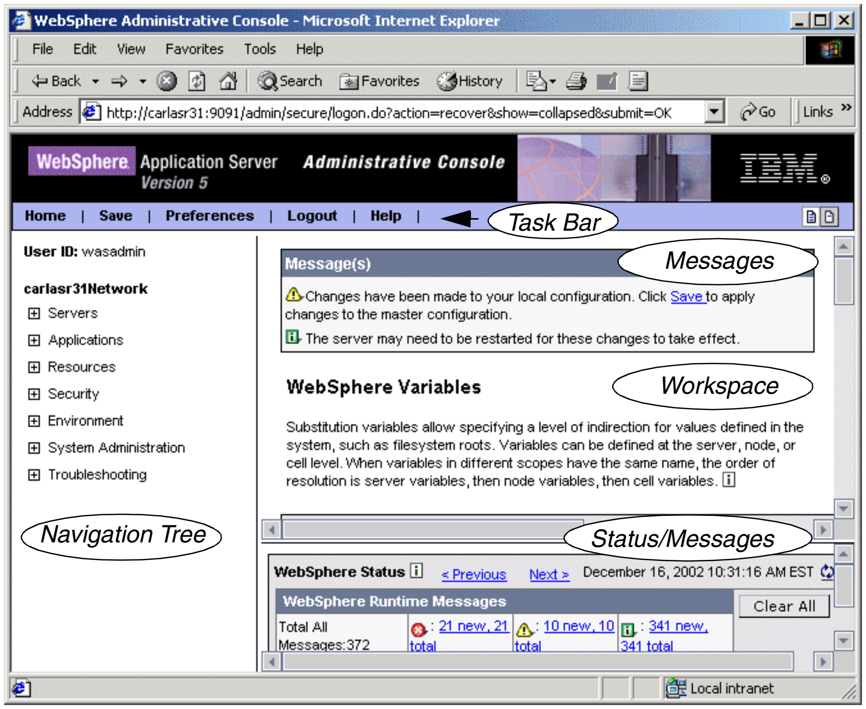
For information on how to change the default location refer to the InfoCenter.

8.3 The graphical interface

The WebSphere administrative console has the following main areas:

- ► Taskbar
- ► Navigation tree
- ► Workspace
- ► Status area

Each area can be resized as desired.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Task Bar
User ID: wasadmin
carlasr31Network
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
Navigation Tree
Message(s)
Messages
Workspace
Status/Messages
< Previous
Next >
Clear All
Local intranet

Changes have been made to your local configuration. Click Save to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

WebSphere Status — December 16, 2002 10:31:16 AM EST

WebSphere Runtime Messages

Total All Messages:372	21 new, 21 total	10 new, 10 total	341 new, 341 total

Figure 8-1 The administrative console graphical Interface

8.3.1 Taskbar

The taskbar is the horizontal bar near the top of the console. It provides the following actions:

- **Home:** Displays the administrative console home page. It contains links to information sources.
- **Save :** Allows you to save pending configuration changes. When you select this you have the opportunity to view the pending changes and save or discard them. A third option, Cancel, simply cancels the save action. It does not discard any changes you made.
- **Preferences:** Allows you to specify several administrative console preferences.
- **Logout:** Logs you out of the administrative console session and displays the Login page. If you have changed the administrative configuration since last saving the configuration to the master repository, the Save page will display before returning to the Login page. Click **Save** to save the changes, **Discard** to return to the administrative console, or **Logout** to exit the session without saving changes.
- **Help:** Opens a new Web browser with detailed online help for the administrative console. (This is not the InfoCenter.)

8.3.2 Navigation tree

The navigation tree on the left side of the console offers links for you to view, select, and manage components in the WebSphere administrative cell.

Clicking a "+" beside a tree folder or item expands the tree for the folder or item. Clicking a "-" collapses the tree for the folder or item. Double-clicking an item toggles its state between expanded and collapsed.

The content displayed on the right side of the console, the *workspace*, depends on the folder or item selected in the tree view.

The following folders are provided for selection:

- **Servers:** Enables configuration of administrative servers, application servers, and clusters.
- **Applications:** Enables installation and management of applications.
- **Resources:** Enables configuration of resources and viewing of information on resources existing in the administrative cell.
- **Security:** Enables configuration and management of WebSphere security and SSL.

- **Environment:** Enables configuration of hosts, Web servers, variables and other components.
- **System Administration:** Enables configuration and management of nodes, cells, console security.
- **Troubleshooting:** Enables you to check for and track configuration errors and problems. Also used to set PMI metrics.

8.3.3 Workspace

The workspace, on the right side of the console in Figure 8-1 on page 262, allows you to work with your administrative configuration after selecting an item from the console navigation tree.

When you click a folder in the tree view, the workspace lists information on instances of that folder type. For example, selecting **Servers -> Application Servers** shows all the application servers configured in this cell. Selecting an item, an application server in this example, will display the Properties page for that item. The Properties page can then be used to view and edit property values.

8.3.4 Status and Messages areas

The Status area displays along the bottom of the console and remains visible as you navigate through the administrative console. The area displays two frames:

- WebSphere Configuration Problems
- WebSphere Runtime Messages

Click **Previous** or **Next** to toggle between the frames. Click the number to view details.

The interval between automatic refreshes can be adjusted by expanding **Preferences** below the messages. In addition, the information displayed can be refreshed at any time by clicking the icon in the upper-right of the area.

The Messages area displays messages relevant to your configuration.

8.4 Using the administrative console

The following sections describe how to use the graphical Web-based administrative console tool to manage the WebSphere Application Server cell.

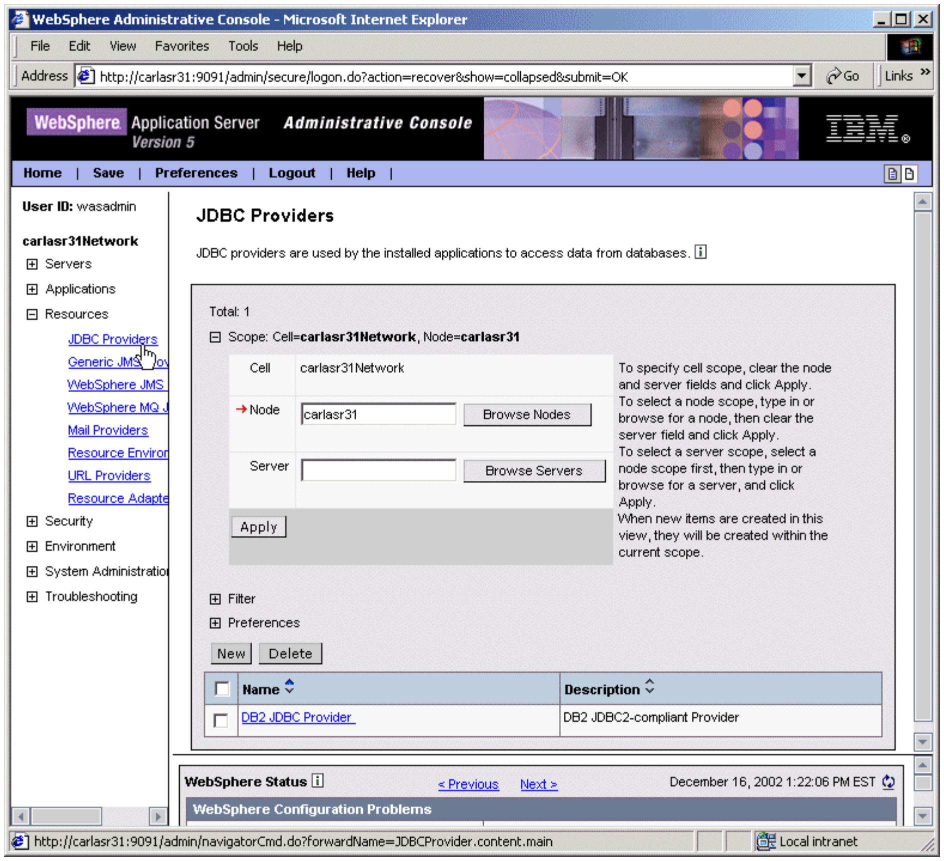
8.4.1 Finding an item

To locate and display items within a cell:

1. Select the associated task from the navigation tree. For example to locate an application server, select **Servers -> Application Servers**.
2. Set the scope to a particular cell, node or server.
3. Set preferences to specify how you would like information to be displayed on the page.

Select task

The navigation tree on the left side of the console contains links to console pages that you use to create and manage components in a WebSphere administrative cell. For example, to create a JDBC provider you would expand **Resources** and then select the **JDBC Providers** action.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
Go
Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Resources
JDBC Providers
Generic JMS
WebSphere JMS
WebSphere MQ J
Mail Providers
Resource Enviror
URL Providers
Resource Adapte
Security
Environment
System Administratio
Troubleshooting
Browse Nodes
Browse Servers
Apply
Filter
Preferences
New
Delete
WebSphere Status
< Previous
Next >
December 16, 2002 1:22:06 PM EST
WebSphere Configuration Problems
http://carlasr31:9091/admin/navigatorCmd.do?forwardName=JDBCProvider.content.main
Local intranet

JDBC Providers

JDBC providers are used by the installed applications to access data from databases.

Total: 1

Scope: Cell=**carlasr31Network**, Node=**carlasr31**

Cell	carlasr31Network
Node	carlasr31
Server	

To specify cell scope, clear the node and server fields and click Apply. To select a node scope, type in or browse for a node, then clear the server field and click Apply. To select a server scope, select a node scope first, then type in or browse for a server, and click Apply. When new items are created in this view, they will be created within the current scope.

	Name	Description
☐	DB2 JDBC Provider	DB2 JDBC2-compliant Provider

Figure 8-2 Working with the administrative console

Select a scope

After selecting an action, use the scope settings to define what information is displayed. Configuration information is defined at three different levels: cell, node, and server.

1. Configurations at the cell level apply to all nodes and servers in the cell. If the node and server fields are blank, the scope is set to the cell level.
2. Configurations at the node level apply to all servers on the node. If a node is specified but the server field is empty, the scope is set to that node.
3. Configurations at the server level apply only to that server. If a server is specified, the scope is set to that server.

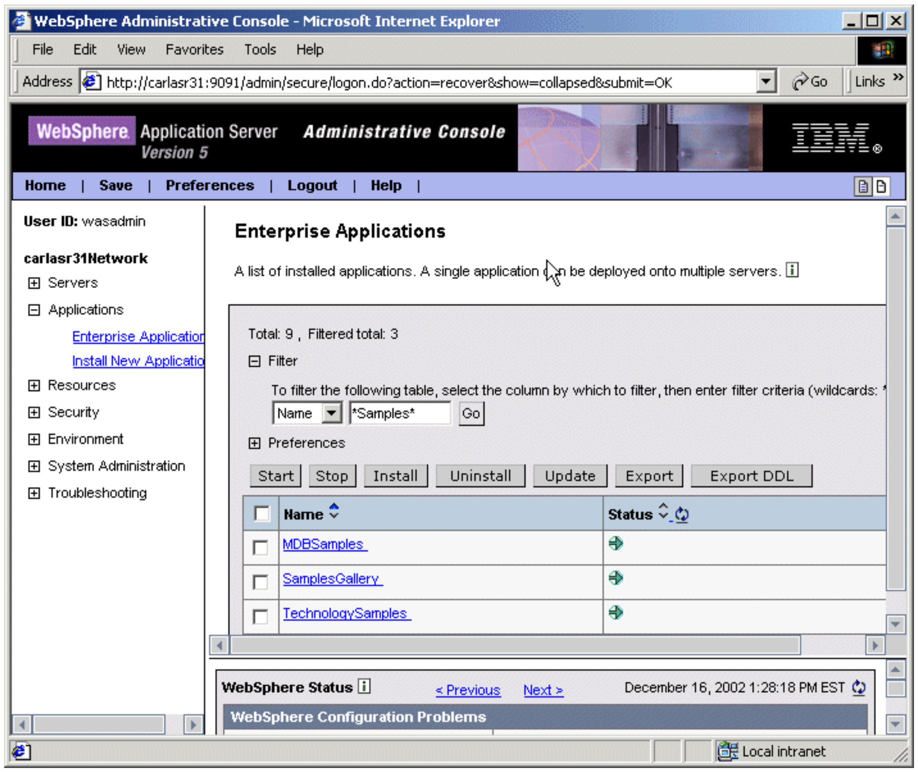
Click **Apply** to set the scope.

The scope setting is available for all resource types, WebSphere variables, shared libraries, and name space bindings.

Set preferences for viewing the console page

After selecting a task and a scope, the administrative console page shows a collection table with all the objects created at that particular scope. For example Figure 8-2 on page 265 shows that there is only one JDBC provider, called DB2 JDBC Provider, created at the node level for node carlasr31. All application servers running on that node can access the DB2 JDBC Provider.

You can filter the contents of the administrative console collection table by using the Filter and Preference settings. For example, in Figure 8-3 on page 267, we selected **Applications -> Enterprise Applications.** Then we used the filter settings to display only those applications that have “Samples” in their name.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Application
Install New Applicatio
Resources
Security
Environment
System Administration
Troubleshooting
Start Stop Install Uninstall Update Export Export DDL
WebSphere Status
< Previous Next >
WebSphere Configuration Problems
Local intranet

Enterprise Applications

A list of installed applications. A single application can be deployed onto multiple servers.

Total: 9 , Filtered total: 3

Filter

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards:

Name *Samples* Go

Preferences

Name	Status
MDBSamples	➔
SamplesGallery	➔
TechnologySamples	➔

December 16, 2002 1:28:18 PM EST

Figure 8-3 Settings that affect how information is displayed on the admin console

The types of characteristics you can filter on will vary depending on the items you are filtering. For example, applications can be filtered by name or by node. JDBC providers can be filtered by name or description.

The Preferences settings allows you to specify the maximum number of rows to display per page and whether to remember search criteria.

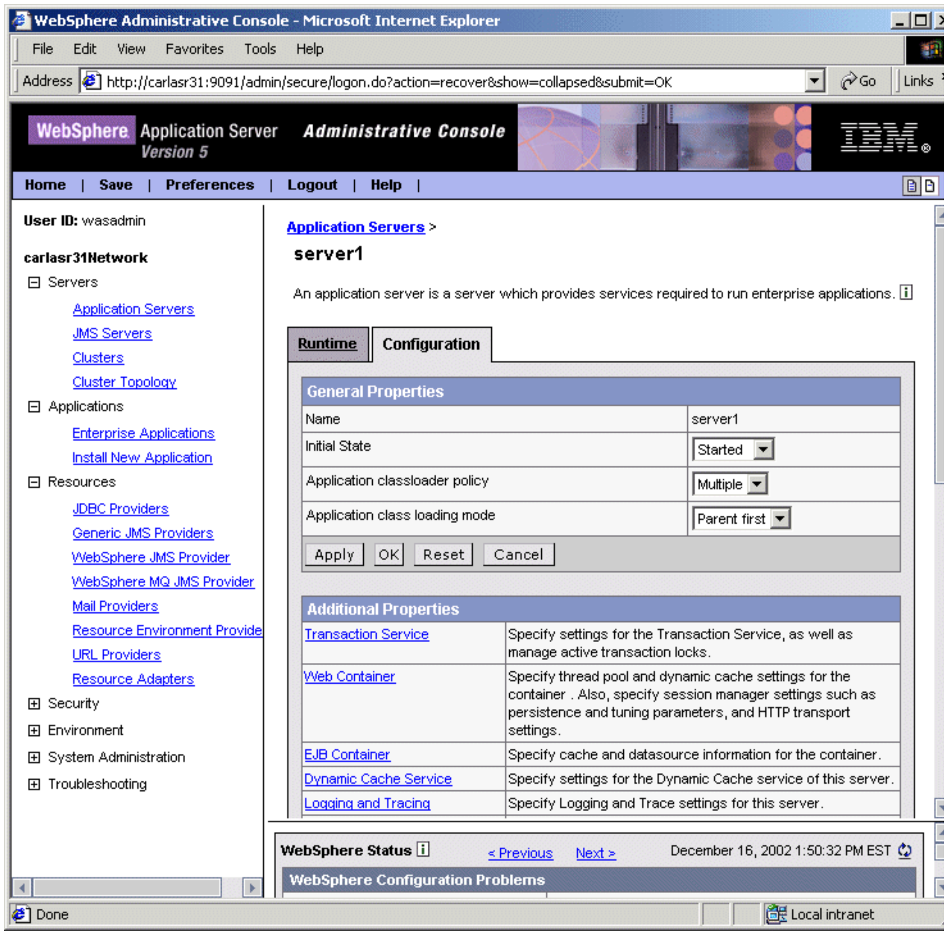
8.4.2 Updating existing items

To edit the properties of an existing item, complete these tasks:

1. Select the category and type in the navigation tree. For example, select **Servers -> Application Servers**.
2. A list of the items of that type in the scope specified will be listed in a collection table in the workspace area. Select an item from the table by clicking it.

3. In some cases you will see a Configuration tab and a Runtime tab. In others you will only see a Configuration tab. Updates are done under the Configuration tab. Specify new properties or edit the properties already configured for that item. The configurable properties will depend on the type of item selected. Often, you will see a General Properties pane and an Additional Properties pane.

   For example, if we click an application server, this open a properties page resembling Figure 8-4.



WebSphere Administrative Console - Microsoft Internet Explorer

File
Edit
View
Favorites
Tools
Help

Address http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK

Go
Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |

User ID: wasadmin

carlasr31Network
- Servers
  - Application Servers
  - JMS Servers
  - Clusters
  - Cluster Topology
- Applications
  - Enterprise Applications
  - Install New Application
- Resources
  - JDBC Providers
  - Generic JMS Providers
  - WebSphere JMS Provider
  - WebSphere MQ JMS Provider
  - Mail Providers
  - Resource Environment Provide
  - URL Providers
  - Resource Adapters
- Security
- Environment
- System Administration
- Troubleshooting

Application Servers >

server1

An application server is a server which provides services required to run enterprise applications.

General Properties	
Name	server1
Initial State	Started
Application classloader policy	Multiple
Application class loading mode	Parent first

Additional Properties	
Transaction Service	Specify settings for the Transaction Service, as well as manage active transaction locks.
Web Container	Specify thread pool and dynamic cache settings for the container . Also, specify session manager settings such as persistence and tuning parameters, and HTTP transport settings.
EJB Container	Specify cache and datasource information for the container.
Dynamic Cache Service	Specify settings for the Dynamic Cache service of this server.
Logging and Tracing	Specify Logging and Trace settings for this server.

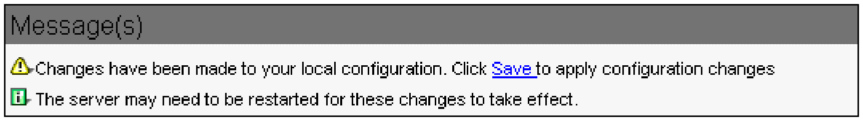
WebSphere Status < Previous Next > December 16, 2002 1:50:32 PM EST

WebSphere Configuration Problems

Figure 8-4 Editing application server properties

The general properties are set directly from this window. Selecting an item in the Additional Properties pane will take you to a new configuration page for those properties.

4. Save changes to the workspace. Click **OK** to save your changes and exit the page or **Apply** to save the changes without exiting. The changes are still temporary. They are only saved to the workspace, not to the master configuration. This still needs to be done.
5. As soon as you save changes to your workspace, you will see a message in the Messages area reminding you that you have unsaved changes.



Changes have been made to your local configuration. Click Save to apply configuration changes

The server may need to be restarted for these changes to take effect.

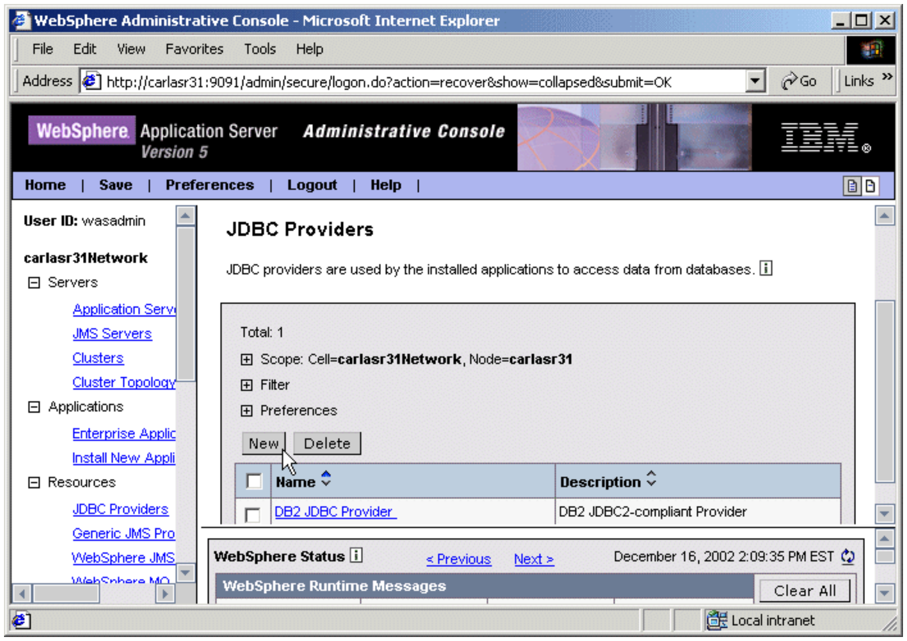
Figure 8-5 Save changes to the master repository

At intervals during your configuration work and at the end you should save the changes to the master configuration. You can do this by clicking **Save** in the message, or by clicking **Save** in the taskbar.

8.4.3 Adding new items

To create new instances of most item types, complete these tasks:

1. Select the category and type in the navigation tree.
2. Select **Scope** and click **Apply** to set it.
3. Click the **New** button above the collection table in the workspace.



WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
Go
Links
WebSphere Application Server Version 5
Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Application Serv
JMS Servers
Clusters
Cluster Topology
Applications
Enterprise Applic
Install New Appli
Resources
JDBC Providers
Generic JMS Pro
WebSphere JMS
WebSphere MQ

JDBC Providers

JDBC providers are used by the installed applications to access data from databases.

Total: 1

Scope: Cell=**carlasr31Network**, Node=**carlasr31**

Filter

Preferences

New Delete

	Name	Description
☐	DB2 JDBC Provider	DB2 JDBC2-compliant Provider

WebSphere Status < Previous Next > December 16, 2002 2:09:35 PM EST

WebSphere Runtime Messages Clear All

Figure 8-6 Create a new item

In general you will be presented with one or more configuration pages in which you have to specify the item properties. The first configuration page is the General Properties page. Fill in the information and click **Apply**.

At this point you may be presented with more configuration options, either in the form of a new configuration page or an Additional Properties pane may appear below the General Properties.

Note: In the configuration pages you can click **Apply** or **OK** to store your changes in the workspace. If you click **OK** you will exit the configuration page. If you click **Apply** you will remain in the configuration page. As you are becoming familiar with the configuration pages, we suggest that you always click **Apply** first. If there are additional properties to configure, you will not see them if you click **OK** and leave the page.

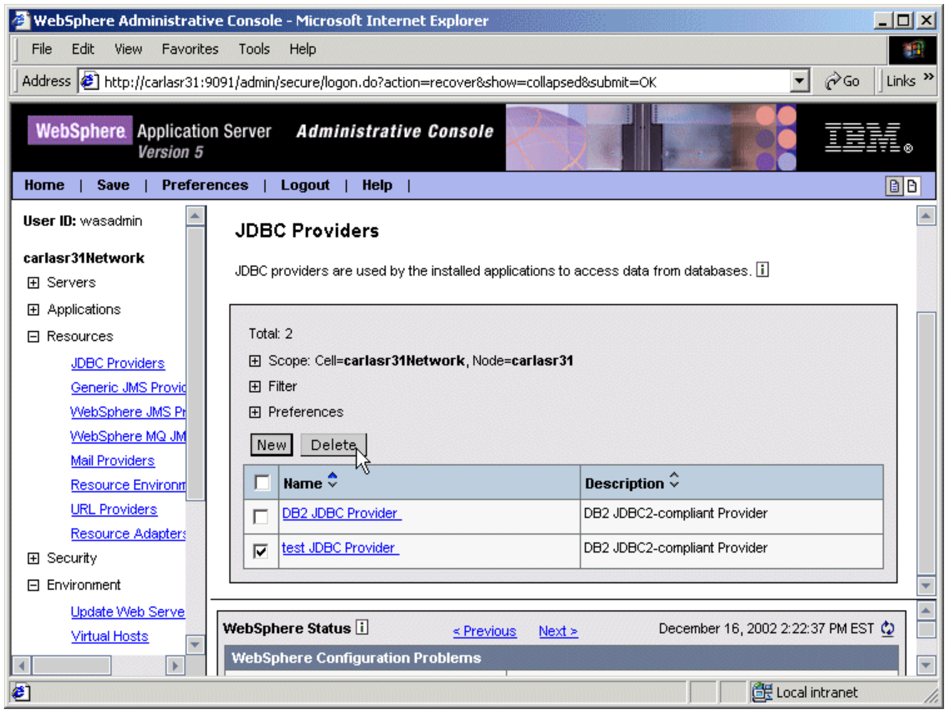
4. Click **Save** in the task bar or in the Messages area when finished.

8.4.4 Removing items

To remove an item, complete these tasks:

1. Find the item.
2. Select the item in the collection table by checking the box next to it.
3. Click **Delete.**
4. If asked whether you want to delete it, click **OK**.
5. Click **Save** to save the changes to the master repository.

For example, to delete an existing JDBC provider, select **Resources -> JDBC Providers**. Check the provider you want to remove and click **Delete**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Resources
JDBC Providers
Generic JMS Provic
WebSphere JMS Pr
WebSphere MQ JM
Mail Providers
Resource Environm
URL Providers
Resource Adapters
Security
Environment
Update Web Serve
Virtual Hosts
New
Delete
WebSphere Status
< Previous
Next >
WebSphere Configuration Problems
Local intranet

JDBC Providers

JDBC providers are used by the installed applications to access data from databases.

Total: 2

Scope: Cell=**carlasr31Network**, Node=**carlasr31**

Filter

Preferences

☐	Name	Description
☐	DB2 JDBC Provider	DB2 JDBC2-compliant Provider
☑	test JDBC Provider	DB2 JDBC2-compliant Provider

December 16, 2002 2:22:37 PM EST

Figure 8-7 Deleting an item

8.4.5 Starting and stopping items

Most items can be started and stopped using the administrative console. To start or stop an item using the console:

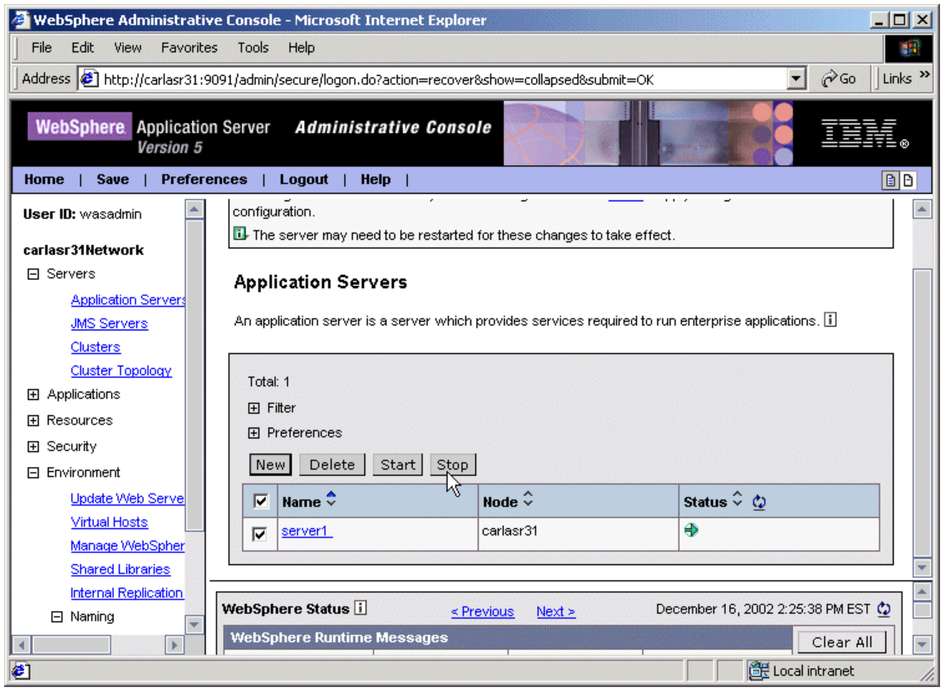
1. Select the item type in the navigation tree.

2. Select the item in the collection table by checking the box next to it.
3. Click **Start** or **Stop.**

The collection table will show the status of the server.

> **Note:** The status of the server can also be “Unavailable”. This will happen when the node agent on the node in which the application server is installed is not active. In this case, the server cannot be started or stopped.

For example, to start an existing application server, select **Servers -> Application Servers.** Place a check mark in the check box beside the application server you want started and click **Start**.



configuration.

The server may need to be restarted for these changes to take effect.

Application Servers

An application server is a server which provides services required to run enterprise applications.

	Name	Node	Status
✓	server1	carlasr31	➔

Figure 8-8 Starting and stopping items

Table 8-1 on page 273 shows how to start/stop the following items.

Table 8-1 How to stop/start items

Type	From	How
Applications	Console	**Applications -> Enterprise Applications**
JMS servers	Console	**Servers -> JMS Servers**
Application servers	Console	**Servers -> Application Servers**
Deployment Manager process (dmgr)	Command prompt	<WAS_ND_HOME>/bin/startManager (.sh) <WAS_ND_HOME>/bin/stopManager (.sh)[2]
	Console[1,2] (stop only)	System Administration> Deployment Manager

[1] Since the Deployment Manager is running the administrative console application, stopping the Deployment Manager from the administrative console will log you out of the current session. Logging in under the same user ID will allow you to save any changes made that were not published to the master configuration repository in the previous session.

[2]Stopping the Deployment Manager doesn't stop any of the node agents or the application servers running under those node agents.

8.4.6 Saving work

As you work with the configuration, your changes are saved to temporary workspace storage. For the configuration changes to take effect, they must be saved to the master configuration and then synchronized (sent) to the nodes. Consider the following:

1. If you work on a page, and click **Apply** or **OK**, the changes will be saved in the workspace under your user ID. This will allow you to recover changes under the same user ID if you exit the session without saving.
2. You need to click **Save** to save changes to the master repository. This can be done from the taskbar, from the Messages area, or when you log in if you logged out without saving the changes.
3. If you don't save changes to the master repository, the changes won't be pushed to your node's configuration repository. Effectively the new settings are lost. They are just available as configuration settings in your temporary workspace.
4. The Save window presents you with the following options:
   - **Save**
   - **Discard**: Discard reverses any changes made during the working session and reverts to the master configuration.

- **Cancel**: Cancel doesn’t reverse changes made during the working session. It just cancels the action of saving to the master repository for now.
- **Synchronize changes with nodes:** This distributes the new configuration to the nodes.

Before deciding whether you want to save or discard changes, you can see what changes will be saved by expanding **View items with changes** in the Save window.

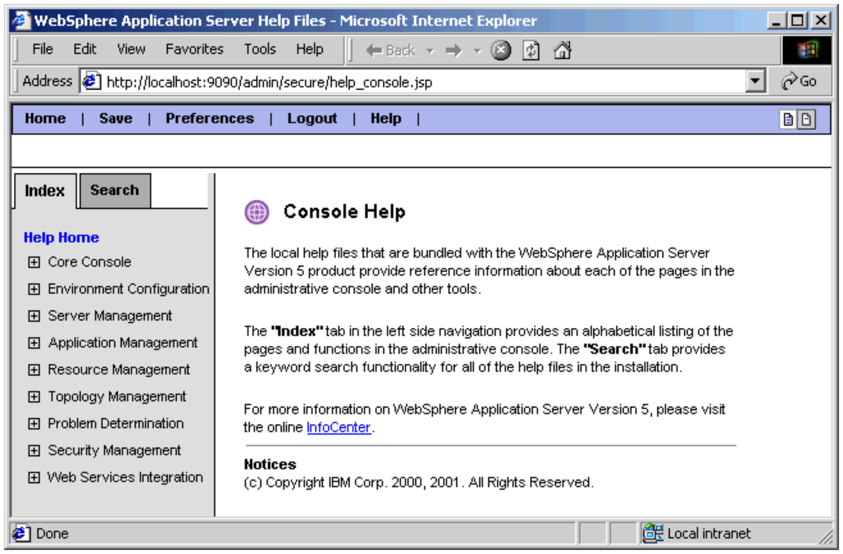
Important: All the changes made during a session are cumulative. Therefore when you decide to save changes to the master repository, either at logon or after clicking **Save** on the taskbar, all changes will be committed. There is no way of being selective about what changes will get saved to the master repository.

5. When you are done, log out of the console using the Logout option on the taskbar.

8.4.7 Getting help

Help is accessible via:

1. The **Help** menu in the taskbar. This opens a new Web browser with online help for the administrative console. It is structured by administrative tasks. See Figure 8-9 on page 275.


WebSphere Application Server Help Files - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost:9090/admin/secure/help_console.jsp
Go
Home | Save | Preferences | Logout | Help |
Index
Search
Help Home
Core Console
Environment Configuration
Server Management
Application Management
Resource Management
Topology Management
Problem Determination
Security Management
Web Services Integration
Done
Local intranet

Console Help

The local help files that are bundled with the WebSphere Application Server Version 5 product provide reference information about each of the pages in the administrative console and other tools.

The **"Index"** tab in the left side navigation provides an alphabetical listing of the pages and functions in the administrative console. The **"Search"** tab provides a keyword search functionality for all of the help files in the installation.

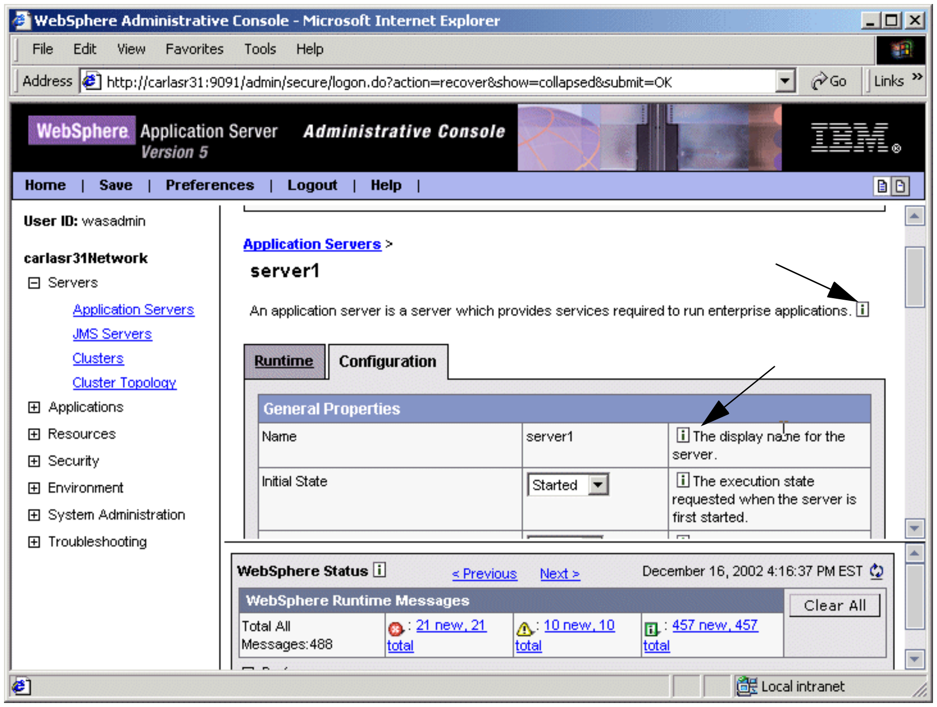
For more information on WebSphere Application Server Version 5, please visit the online InfoCenter.

Notices
(c) Copyright IBM Corp. 2000, 2001. All Rights Reserved.

Figure 8-9 Online help

2. The **Hide Field and Page Descriptions toggle**. When disabled, console pages will show an "i" icon at the top of the workspace for page descriptions, and beside a field to see information just about that particular item. Click it to access description information.

   For example, Figure 8-10 on page 276 shows that there is description information available at the page level and field levels. This will just be a subset of the information contained at the page level.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=recover&show=collapsed&submit=OK
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Application Servers
JMS Servers
Clusters
Cluster Topology
Applications
Resources
Security
Environment
System Administration
Troubleshooting
Application Servers >
server1
Runtime
Configuration
General Properties
WebSphere Status
< Previous
Next >
December 16, 2002 4:16:37 PM EST
WebSphere Runtime Messages
Clear All
Local intranet

An application server is a server which provides services required to run enterprise applications.

General Properties		
Name	server1	The display name for the server.
Initial State	Started	The execution state requested when the server is first started.

Total All Messages:488	: 21 new, 21 total	: 10 new, 10 total	: 457 new, 457 total

Figure 8-10 Description information

3. The InfoCenter can be viewed online or downloaded from:

   `http://www.ibm.com/software/webservers/appserv/infocenter.html`

8.5 Securing the administrative console

WebSphere Application Server V5 provides the ability to secure the administrative console so only authenticated users can use it. In order to take advantage of this feature, you will need to first activate WebSphere global security. Enabling security is an important step in ensuring a safe WebSphere environment. However, the considerations and decisions that you must make before you do this are outside the scope of this book. To understand the security options and for help designing a secure system, refer to *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

This section assumes that you have enabled WebSphere global security and therefore concentrates on the steps needed to secure the console.

Console security is based on identifying users or groups that are defined in the active user registry and assigning roles to those user. When a user logs in to the administrative console, a valid administrator user ID and password must be entered. The roles determine the administrative actions the user can take.

Users and groups are added and roles assigned to them by selecting **System Administration -> Console Users** or **System Administration -> Console Groups**.

The roles available are:

- **Monitor**: Allows a user to view the WebSphere configuration and current state.
- **Configurator**: Monitor privilege plus the ability to change the WebSphere configuration.
- **Operator**: Monitor privilege plus the ability to change the runtime state, such as starting and stopping services.
- **Administrator**: Operator plus configurator properties.

After saving the configuration, you must restart the WebSphere Application Server for a base installation or the Deployment Manager in a Network Deployment environment.

Note: In a Network Deployment environment, all the application servers will also need to be restarted.

The next time you log in to the administrative console, you must authenticate with one of the users that were identified having an administrative role.

8.6 Common administrative tasks

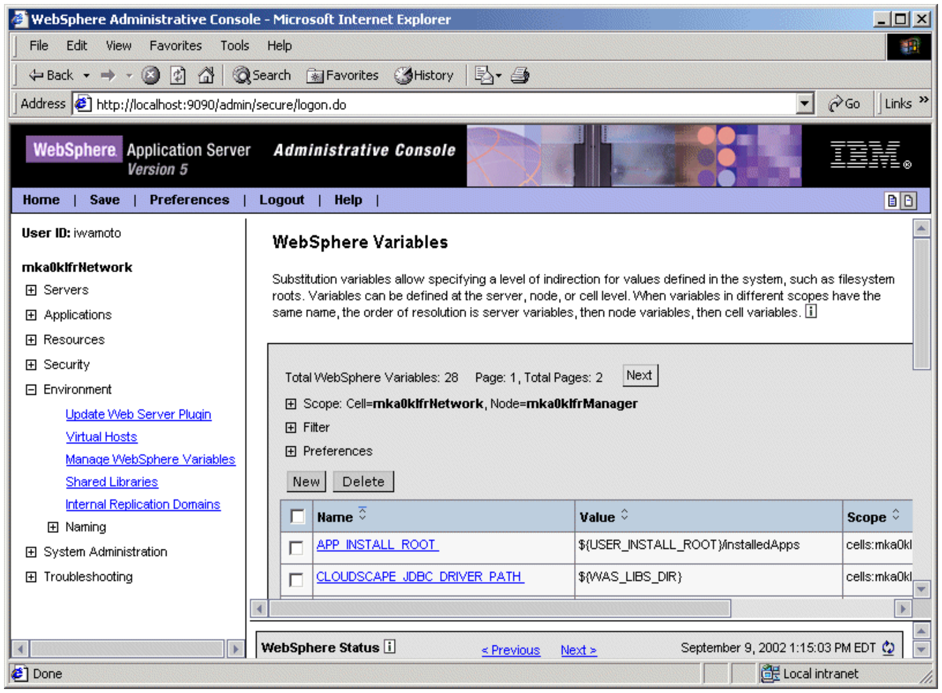
This section will take a look at how to perform some of the more common administrative tasks using the administrative console and commands. For information about using the wsadmin tool to do the same tasks, see Chapter 16, “Command-line administration and scripting” on page 779.

8.6.1 Using variables

WebSphere variables are name/value pairs used to represent variables used in the configuration files. This makes it easier to manage a large configuration.

To set a WebSphere variable:

1. Click **Environment -> Manage WebSphere Variables** and click **New**.



WebSphere Administrative Console - Microsoft Internet Explorer

Address http://localhost:9090/admin/secure/logon.do

User ID: iwamoto

mka0klfrNetwork

- ⊞ Servers
- ⊞ Applications
- ⊞ Resources
- ⊞ Security
- ⊟ Environment
  - Update Web Server Plugin
  - Virtual Hosts
  - Manage WebSphere Variables
  - Shared Libraries
  - Internal Replication Domains
  - ⊞ Naming
- ⊞ System Administration
- ⊞ Troubleshooting

WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Total WebSphere Variables: 28 Page: 1, Total Pages: 2 Next

⊞ Scope: Cell=**mka0klfrNetwork**, Node=**mka0klfrManager**

⊞ Filter

⊞ Preferences

New Delete

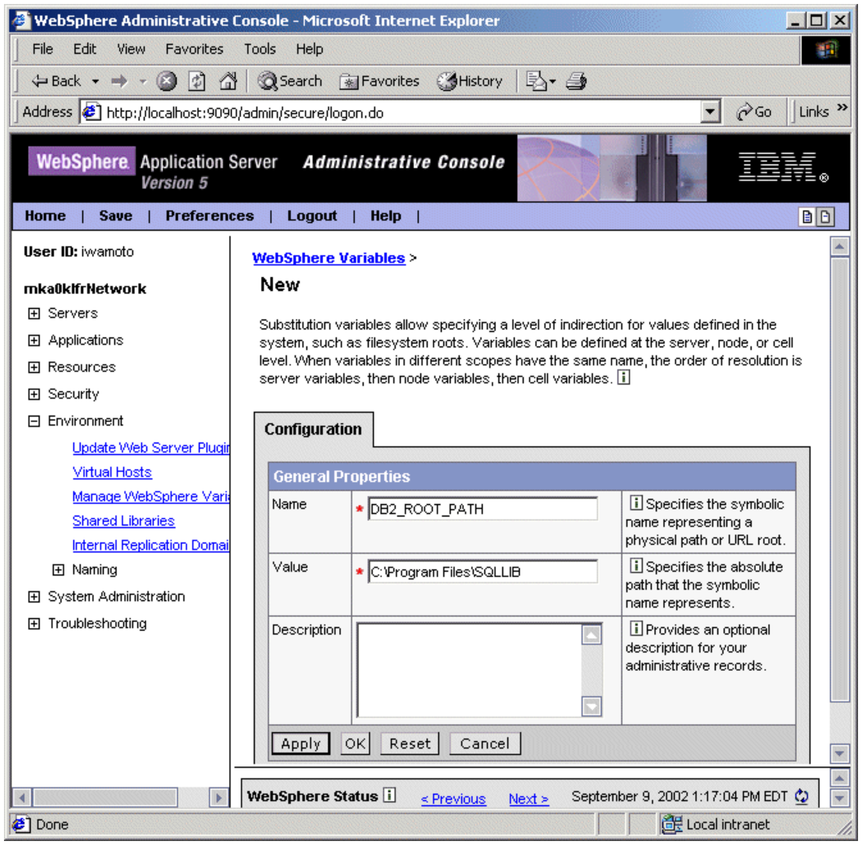
	Name	Value	Scope
☐	APP_INSTALL_ROOT	${USER_INSTALL_ROOT}/installedApps	cells:mka0kl
☐	CLOUDSCAPE_JDBC_DRIVER_PATH	${WAS_LIBS_DIR}	cells:mka0kl

WebSphere Status < Previous Next > September 9, 2002 1:15:03 PM EDT

Done Local intranet

Figure 8-11 WebSphere variables

2. Enter a name and value and click **Apply**. In this example, DB2_ROOT_PATH is used as the name and c:\Program Files\SQLLIB is used as the value.


WebSphere Variables > New

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Configuration

General Properties

Name	* DB2_ROOT_PATH	Specifies the symbolic name representing a physical path or URL root.
Value	* C:\Program Files\SQLLIB	Specifies the absolute path that the symbolic name represents.
Description		Provides an optional description for your administrative records.

Figure 8-12 New WebSphere variable

8.6.2 Managing nodes

Managing nodes is a concept specific to a Network Deployment environment. Nodes are managed by the Deployment Manager through a process known as a node agent that resides on each node. In order to manage a node in a Network Deployment environment, the node must be defined and the node agent on each WebSphere Application Server node must be started.

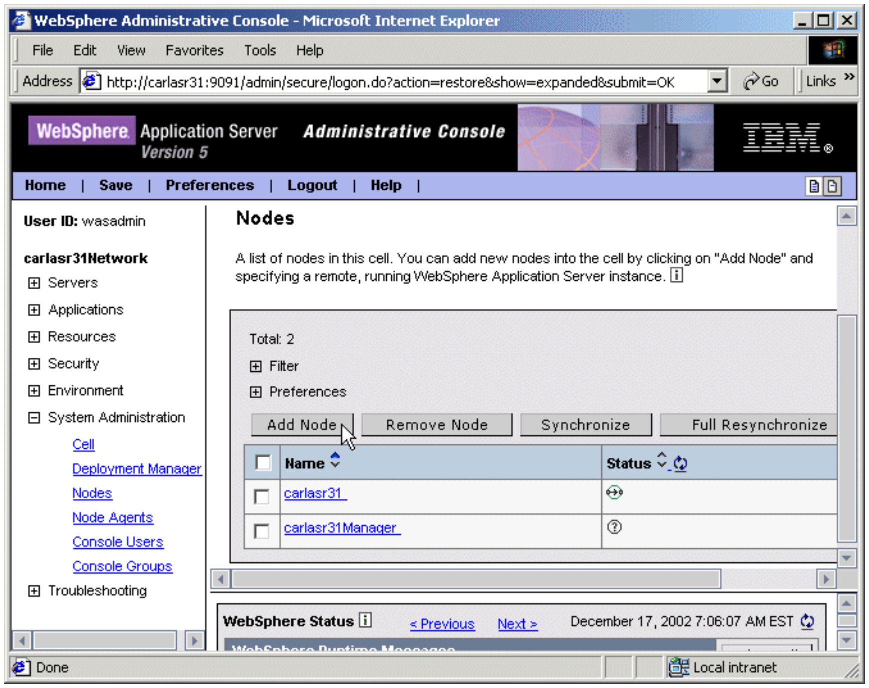
Adding a node

There are two ways of incorporating a stand-alone application server installation (node) into an existing administration cell.

Method 1

From the Network Deployment administrative console:

1. Select **System Administration -> Nodes -> Add Node**.
2. Specify the host name of the node to be added to the cell.



A list of nodes in this cell. You can add new nodes into the cell by clicking on "Add Node" and specifying a remote, running WebSphere Application Server instance.

	Name	Status
☐	carlasr31	
☐	carlasr31Manager	

December 17, 2002 7:06:07 AM EST

Figure 8-13 Working with nodes

Method 2

From the command line:

1. Change directory to the bin directory of the stand-alone application server installation.
2. Run the **`addnode`** command. See A.5, “addNode” on page 843 for information on this command.

Note: By default `addNode` does not carry over applications from the stand-alone servers to the cell. So after adding a node you get a node agent, a JMS Server, any application servers and their configurations, but not the applications.

If you want the applications to be carried over from the servers on the node into the cell, use the `addnode` program with the "`-includeapps`" option.

For example to add the node installed on node1 to the Network Deployment cell on host "dmgrnode", enter the following from node1:

```
cd c:\websphere\appserver\bin
addnode dmgrnode 8879 -includeapps
```

The port specified, 8879, is the default SOAP connector port for the Deployment Manager. To verify the port:

1. Select **System Administration -> Deployment Manager**.
2. Click **End Points**.
3. Select the **SOAP_CONNECTOR_ADDRESS** to see the port.

Removing a node

There are two ways of removing a node from a network distributed administration cell.

Method 1

From the administrative console:

1. Select **System Administration -> Nodes**.
2. Place a check mark in the check box beside the node you want to remove and click **Remove Node**.

Method 2

From the command line:

1. Change directory to the bin directory of the base application server installation for that node.
2. Run `removeNode`. See A.6, "removeNode" on page 847 for information on this command.

Note: To avoid loss of configuration information when a node is detached from a cell, modify the application servers before removing the node. Click **Servers -> Application Servers** in the console navigation tree. Select the application server on the node you are going to remove. Select **Administrative Services** and check the **Standalone** box. This option is also available at the node agent level. Save the configuration and synchronize.

Adjusting node agent synchronization

Configuration synchronization between the node and the Deployment Manager is enabled by default. During a synchronization operation, a node agent checks with the Deployment Manager to see if any configuration documents that apply to the node have been updated. New or updated documents are copied to the node repository, and deleted documents are removed from the node repository. The interval between synchronizations is configurable via the administrative console:

1. Expand **System Administration -> Node Agents** in the administrative console.
2. Select the node agent process on the appropriate server to open the properties page.
3. In Additional Properties, click **File Synchronization Service**.
4. Configure the synchronization interval. By default the synchronization interval is set to one minute.

Explicit synchronization can be forced by selecting **System Administration -> Nodes.** Select a node and click **Synchronize** or **Full Synchronization.** Synchronize performs an immediate synchronization on the selected node using the optimization settings for the file synchronization service of the node agent.

The Full Synchronization option disregards any synchronization optimization settings and ensures that the node and cell configuration are identical.

Tip: Increasing the synchronization interval in a production environment is recommended to reduce the overhead.

Starting and stopping nodes

A node consists of the node agent and the servers. There are several ways to start and stop a node and/or node agent. Before using any of these methods, be sure to note whether it affects the entire node, including servers, or just the node agent.

Starting a node agent
When a node agent is stopped, the Deployment Manager has no way to communicate with it. Therefore, the node agent has to be started from a node. From a command prompt on the node:

- Windows: **`<WAS_HOME>\bin\startNode`**
- UNIX: **`<WAS_HOME>\bin\startNode.sh`**

Stopping a node agent
To stop the node agent but leave the servers running:

1. From the administrative console, select **System Administration -> Node Agents**.
2. Check the box beside the node agent for the server and click **Stop**.

Or, from a command prompt:

- Windows: **`<WAS_HOME>\bin\stopNode`**
- UNIX: **`<WAS_HOME>\bin\stopNode.sh`**

Stopping a node (stops the node agent and servers)
A node can be stopped from the Network Deployment administrative console or from a node command prompt.

To stop a node and all the servers on the node:

1. From the administrative console, select **System Administration -> Nodes**.
2. Check the box beside the node and click **Stop**.

Restarting a node agent
1. From the administrative console, select **System Administration -> Node Agents**.
2. Check the box beside the node agent for the server and click **Restart**.

8.6.3 Managing application servers

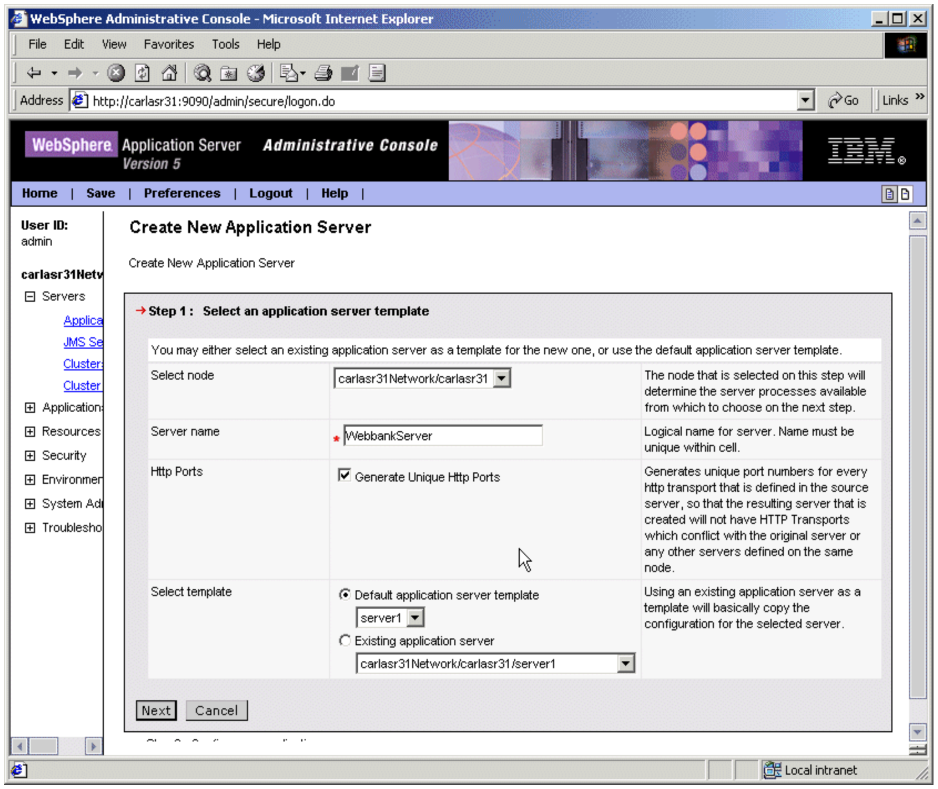
This section describes how to perform common administration tasks on application servers using the WebSphere administrative console.

Creating an application server
By default, every WebSphere Application Server node has one application server called “server1”. This application server hosts the sample applications, and in a base WebSphere environment, it also hosts the administrative console application and provides the JMS server functions. Each application server runs

in its own JVM and as such, may have limitations related to performance. In this section, we discuss how to create and configure application servers.

To create a server:

1. Select **Servers -> Application Servers**. A list of application servers defined in the cell will appear in the workspace.
2. Click **New**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Netw
Servers
Applica
JMS Se
Cluster
Cluster
Application
Resources
Security
Environmer
System Adi
Troublesho
Next Cancel
Local intranet

Create New Application Server

Create New Application Server

→ Step 1 : Select an application server template

You may either select an existing application server as a template for the new one, or use the default application server template.

Select node	carlasr31Network/carlasr31	The node that is selected on this step will determine the server processes available from which to choose on the next step.
Server name	* WebbankServer	Logical name for server. Name must be unique within cell.
Http Ports	✓ Generate Unique Http Ports	Generates unique port numbers for every http transport that is defined in the source server, so that the resulting server that is created will not have HTTP Transports which conflict with the original server or any other servers defined on the same node.
Select template	● Default application server template server1 ○ Existing application server carlasr31Network/carlasr31/server1	Using an existing application server as a template will basically copy the configuration for the selected server.

Figure 8-14 Creating a new application server

- **Select Node:** Select the node for the application server from the drop-down list.
- **Server name:** Type in a name for the server. It must be unique within the cell.
- **Http Ports**: Checking the **Generate Unique Http Ports** check box will ensure a unique HTTP port for the Web container is defined. Otherwise, the ports are copied from the template server. Copying ports can be useful for creating identical application servers on different nodes.

   - **Select template:** Select an existing application server to use as a template for creating this server.

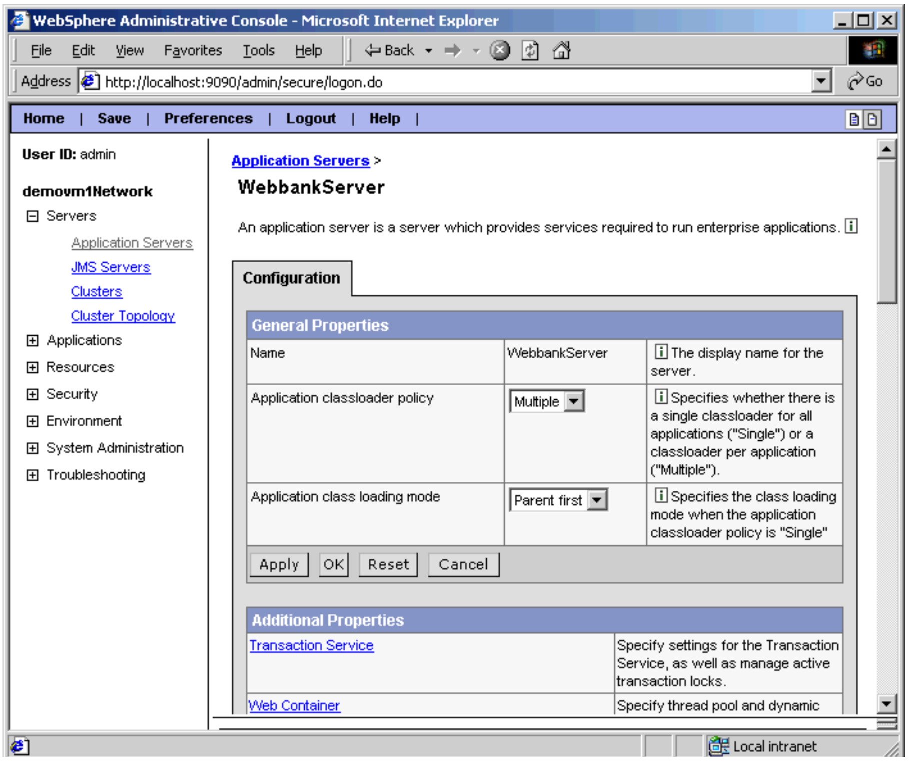
3. Click **Next**. A Summary window will outline what action will be taken and any issues that might result.
4. Click **Finish** to create the application server.

Application server configuration parameters

When you create a new application server, it inherits most of its configuration settings from the template server specified.

To view or modify these settings, select **Servers -> Application Servers**. A list of application servers defined in the cell will appear in the workspace. Click the name of the application server to make a modification. The server properties appear in two tables: the General Properties table and the Additional Properties table.

There are many configuration parameters associated with a server, far too many to cover in detail here. The InfoCenter and administrative console contain information about these settings. This section will give you a quick overview of the types of settings you will find.



WebbankServer

An application server is a server which provides services required to run enterprise applications.

General Properties

Name	WebbankServer	The display name for the server.
Application classloader policy	Multiple	Specifies whether there is a single classloader for all applications ("Single") or a classloader per application ("Multiple").
Application class loading mode	Parent first	Specifies the class loading mode when the application classloader policy is "Single"

Additional Properties

Transaction Service	Specify settings for the Transaction Service, as well as manage active transaction locks.
Web Container	Specify thread pool and dynamic

Figure 8-15 Application server configuration

General properties

The general properties consist of a few items which you can see immediately.

- **Application classloader policy:** Application classloaders consist of EJB modules, dependency JAR files, resource adapters, and shared libraries. This selection will specify if any classloader isolation is to take place between applications running on this application server.

  If **SINGLE** is selected, only one object is created for managing loading of all EJB modules, dependency JAR files, resource adapters, and shared files required by the applications on the application server. This has the benefit of low memory usage, since shared classes are only loaded once. The drawback is that all applications must agree on a common version of all classes.

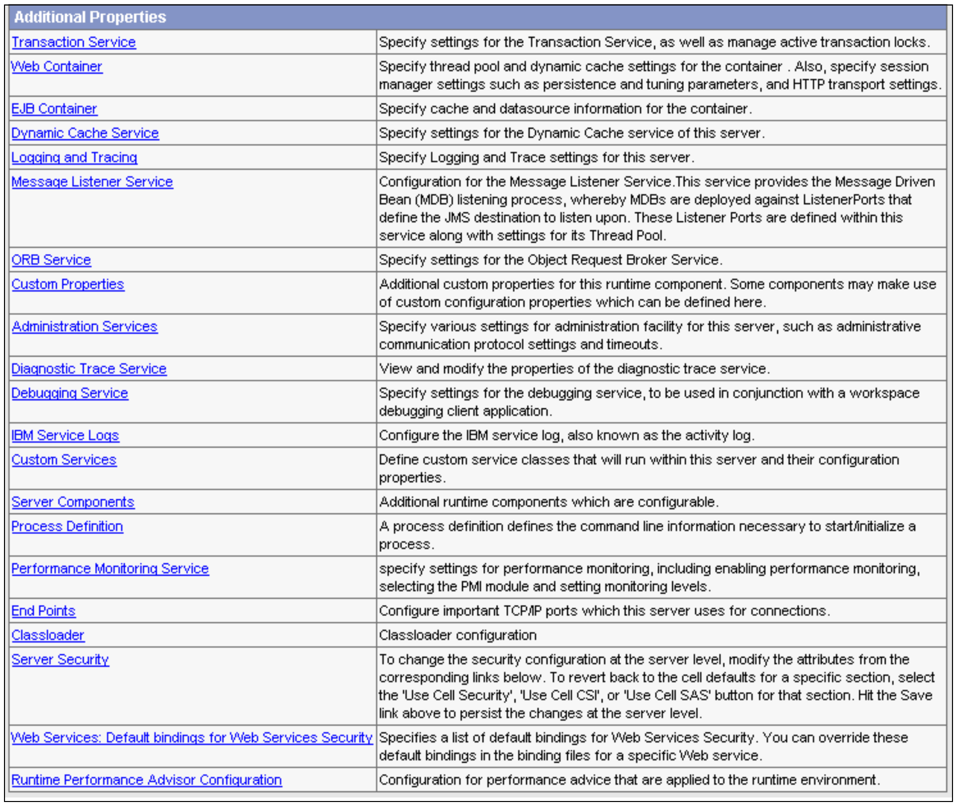
  Choosing **MULTIPLE** creates a classloader object for every application in the application server, isolating applications classes (EJB modules, dependency JAR files, resource adapters, and shared libraries) from one another.

Note: The WAR classloader policy for Web modules is specified on a per application basis. The application classloader policy configured at the application server level only affects EJB modules, dependency JAR files, resource adapters, and shared libraries for applications residing on this application server.

- **Application class loading mode:** The class loading mode allows overriding the delegation in which classes are located and loaded. The standard J2EE classloader scheme is **PARENT_FIRST**, in which an application first delegates classloading to its parent classloader before trying to locate and load a class itself. The **PARENT_LAST** reverses the J2EE standard classloading behavior and makes any classloader try to locate and load a class before delegation takes place. The PARENT_LAST policy allows an application classloader to override and provide its own version of a class that exists in the parent classloader.

Additional properties

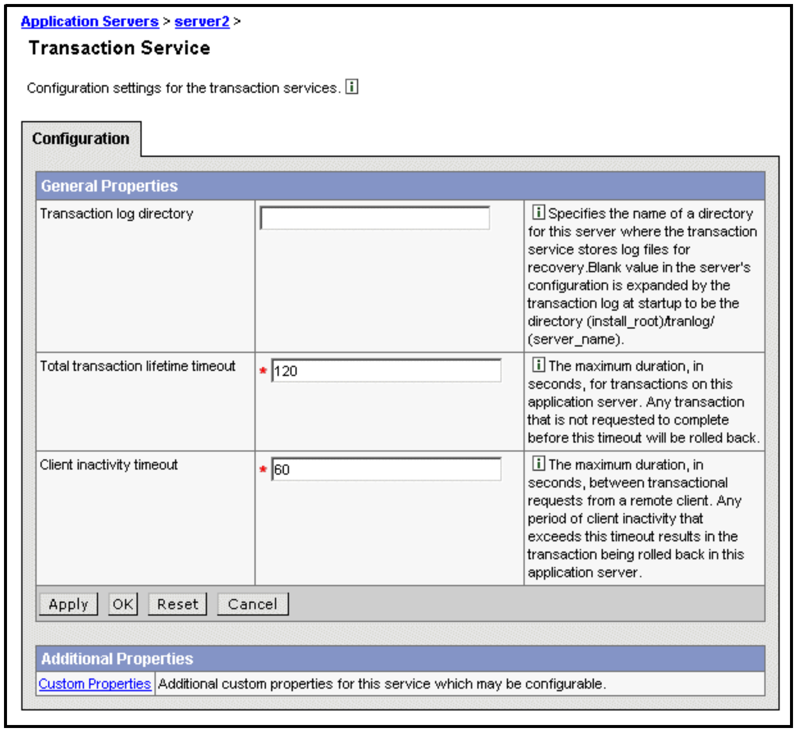
The items in the Additional Properties table consist of categories of properties. To see or modify these properties, you will need to click each relevant category.



Additional Properties	
Transaction Service	Specify settings for the Transaction Service, as well as manage active transaction locks.
Web Container	Specify thread pool and dynamic cache settings for the container . Also, specify session manager settings such as persistence and tuning parameters, and HTTP transport settings.
EJB Container	Specify cache and datasource information for the container.
Dynamic Cache Service	Specify settings for the Dynamic Cache service of this server.
Logging and Tracing	Specify Logging and Trace settings for this server.
Message Listener Service	Configuration for the Message Listener Service.This service provides the Message Driven Bean (MDB) listening process, whereby MDBs are deployed against ListenerPorts that define the JMS destination to listen upon. These Listener Ports are defined within this service along with settings for its Thread Pool.
ORB Service	Specify settings for the Object Request Broker Service.
Custom Properties	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.
Administration Services	Specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts.
Diagnostic Trace Service	View and modify the properties of the diagnostic trace service.
Debugging Service	Specify settings for the debugging service, to be used in conjunction with a workspace debugging client application.
IBM Service Logs	Configure the IBM service log, also known as the activity log.
Custom Services	Define custom service classes that will run within this server and their configuration properties.
Server Components	Additional runtime components which are configurable.
Process Definition	A process definition defines the command line information necessary to start/initialize a process.
Performance Monitoring Service	specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.
End Points	Configure important TCP/IP ports which this server uses for connections.
Classloader	Classloader configuration
Server Security	To change the security configuration at the server level, modify the attributes from the corresponding links below. To revert back to the cell defaults for a specific section, select the 'Use Cell Security', 'Use Cell CSI', or 'Use Cell SAS' button for that section. Hit the Save link above to persist the changes at the server level.
Web Services: Default bindings for Web Services Security	Specifies a list of default bindings for Web Services Security. You can override these default bindings in the binding files for a specific Web service.
Runtime Performance Advisor Configuration	Configuration for performance advice that are applied to the runtime environment.

Figure 8-16 Additional Properties for the Application Server

- **Transaction service properties:** The transaction service properties allow you to specify settings for the transaction service, as well as manage active transaction locks.


Application Servers > server2 >

Transaction Service

Configuration settings for the transaction services.

Configuration

General Properties		
Transaction log directory		Specifies the name of a directory for this server where the transaction service stores log files for recovery.Blank value in the server's configuration is expanded by the transaction log at startup to be the directory (install_root)/tranlog/(server_name).
Total transaction lifetime timeout	* 120	The maximum duration, in seconds, for transactions on this application server. Any transaction that is not requested to complete before this timeout will be rolled back.
Client inactivity timeout	* 60	The maximum duration, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server.

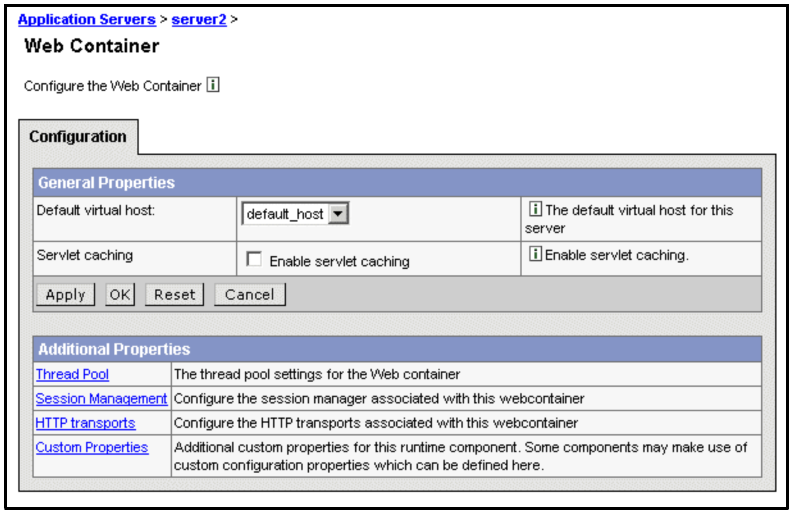
Additional Properties	
Custom Properties	Additional custom properties for this service which may be configurable.

Figure 8-17 Transaction service settings

- **Transaction Log Directory:** Specifies the directory location for the transaction service on the application server to store log files for recovery. If no value is provided for the location of this file, its location defaults to (install_root)/tranlog/(server_name) at server startup.
- **Total Transaction Lifetime Timeout:** The maximum duration, in seconds, for transactions on this application server to complete. Any transaction that is not completed before the timeout will be rolled back by the application server. The default value for the transaction timeout is 120 seconds. As a best practice you will want to examine the types of transactions you will be monitoring and determine the total time it should take to complete these transactions. Existing business intelligence may help with this decision.
- **Client Inactivity Timeout:** The maximum delay, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back by the application server.

When the application server is running, a Runtime tab is available in the Transaction Service properties workspace. From here, running transactions can be managed and timeout settings can be modified at runtime.

- **Web container properties:** The Web container page allows you to specify thread pool and dynamic cache settings for the container as well as specify session manager settings such as persistence and tuning parameters, and HTTP transport properties. The Web container will serve application requests for servlets and JSPs.


Application Servers > server2 >

Web Container

Configure the Web Container

Configuration

General Properties		
Default virtual host:	default_host	The default virtual host for this server
Servlet caching	☐ Enable servlet caching	Enable servlet caching.

Additional Properties	
Thread Pool	The thread pool settings for the Web container
Session Management	Configure the session manager associated with this webcontainer
HTTP transports	Configure the HTTP transports associated with this webcontainer
Custom Properties	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.

Figure 8-18 Web container attributes

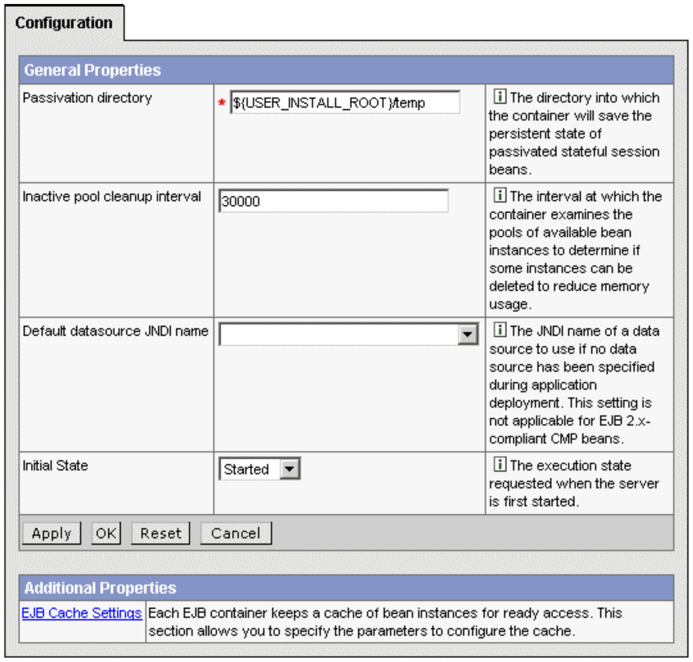
 - **Thread Pool:** The thread pool specifies the possible maximum number of concurrently running threads in the Web container. As one thread is needed for every client request, this directly relates to the number of active clients that can possibly access the Web container on this application server at any given time. A timeout value can be specified for the application server to remove threads from the pool based on a timed period of inactivity.

   Finally an option for creating threads beyond the maximum pool size is available. Be careful when using this option. It can have the unexpected effect of allowing the Web container to create more threads than the JVM might be able to process, creating a resource shortage and bringing the application server to a halt.

   For information on thread pools and determining their size, see *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.

  - **Session Management:** Allows you to determine how the Web container will manage conversational data from the client. The following options are configurable:
    - **Session tracking mechanism**: you have the option of enabling SSL ID tracking, enabling cookies, and enabling URL rewriting. There is also the ability to enable protocol switch rewriting. This option allows the Web container to receive requests over HTTP and then redirect them to HTTPS.
    - **Maximum in memory session count:** Specifies the maximum number of sessions to maintain in memory. Spawning of sessions on the Web container could continue until the Web container runs out of memory. By setting this limit, you eliminate the possibility of having an infinite amount of sessions created in the Web container. This also identifies the amount of sessions in memory and the same number of sessions will be held persistently in the case of an application server failure.
    - **Overflow:** Specifies whether to allow the number of sessions in memory to exceed the value specified by Max In Memory Session Count property. This is valid only in non-persistent sessions mode, when you have decided not to use the option for persistent sessions.
  - **HTTP Transports:** Allows you to add to or configure the transports associated with the Web container. WebSphere maintains a counter for HTTP transport ports. The counter starts at port 9080 for HTTP and 9443 HTTPS. When this option is selected, and a new application server is created, WebSphere assigns the next port number available to the Web container HTTP transports, for example 9081 for HTTP and 9444 for HTTPS.

    Port numbers must be unique for each application server instance on a given machine.
  - **Custom Properties:** Allows you to specify name/value pairs for configuring internal system properties. Some components may make use of custom configuration properties, which can be defined here. It is not common to pass information to the Web container this way, but the J2EE specification indicates this as a requirement. Most configuration information can be handled programmatically or through the deployment descriptor.
- **EJB container properties:** These properties allow you configure the services provided by the EJB container.


Configuration

General Properties

Passivation directory	* ${USER_INSTALL_ROOT}/temp	The directory into which the container will save the persistent state of passivated stateful session beans.
Inactive pool cleanup interval	30000	The interval at which the container examines the pools of available bean instances to determine if some instances can be deleted to reduce memory usage.
Default datasource JNDI name		The JNDI name of a data source to use if no data source has been specified during application deployment. This setting is not applicable for EJB 2.x-compliant CMP beans.
Initial State	Started	The execution state requested when the server is first started.

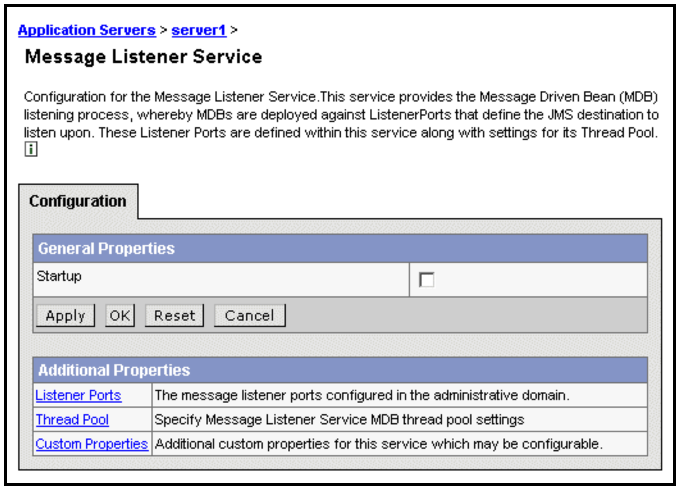
Additional Properties

EJB Cache Settings	Each EJB container keeps a cache of bean instances for ready access. This section allows you to specify the parameters to configure the cache.

Figure 8-19 EJB container settings

- **Passivation Directory:** This attribute provides the directory that you can use to store the persistent state of passivated, stateful session EJBs. If you are using the EJB container to manage session data, you should give WebSphere the ability to swap data to disk when necessary. This directory will tell WebSphere where to hold EJB session data when it passivates and activates beans from the pool.
- **Inactive pool cleanup interval:** Because WebSphere will build a pool of EJBs to satisfy incoming requests, we need to tell it when to remove beans from this pool to preserve resources. This attribute allows you to define the interval at which the container examines the pools of available bean instances to determine if some instances can be deleted to reduce memory usage.
- **Default datasource JNDI name:** Here you can set a default data source to use for EJBs that have no individual data source defined. This setting is not applicable for EJB 2.x-compliant CMP beans.
- **Initial state:** This attribute allows you to identify the state of the container when WebSphere is started. If you have to recycle the application server, this attribute will be used to determine whether to start the EJB container at that time, or wait for a manual start.

  - **EJB Cache settings:** You can set up two types of cache settings in WebSphere:
    - **Cleanup interval:** This attribute allows you to set the interval at which the container attempts to remove unused items from the cache in order to reduce the total number of items in cache to the value we set in the cache size attribute.
    - **Cache size:** This attribute specifies the number of buckets in the active instance list within the EJB container. This attribute will be used by WebSphere to determine how large the cache will be and when to remove components from the cache to reduce its size.
- **Dynamic cache service:** This page allows you to specify settings for the dynamic cache service of this server. If you are planning to use the dynamic caching service, please refer to *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.
- **Logging and tracing:** This page allows you to specify settings for diagnostic trace, JVM logs, IBM service logs, and process logs. For information about setting up and using these diagnostic procedures, see Chapter 15, "Troubleshooting" on page 715.
- **Message listener service:** The message listener service provides support for message-driven beans. Before configuring these properties, review the information in Chapter 11, "Asynchronous messaging" on page 451. In particular, read the following sections which discuss the message listener service:
  - 11.1.4, "Message-driven beans" on page 457.
  - 11.2.3, "WebSphere support for message-driven beans" on page 463.


Application Servers > server1 >

Message Listener Service

Configuration for the Message Listener Service.This service provides the Message Driven Bean (MDB) listening process, whereby MDBs are deployed against ListenerPorts that define the JMS destination to listen upon. These Listener Ports are defined within this service along with settings for its Thread Pool.

Configuration

General Properties

Startup	☐

Apply | OK | Reset | Cancel

Additional Properties

Listener Ports	The message listener ports configured in the administrative domain.
Thread Pool	Specify Message Listener Service MDB thread pool settings
Custom Properties	Additional custom properties for this service which may be configurable.

Figure 8-20 Message listener service

This page allows you to define:

- **Listener ports:** Each port specifies the JMS connection factory and JMS destination that an MDB, deployed against that port, will listen on.
- **Thread pools:** This option allows the administrator to create a thread pool that will be available for use by the beans at runtime. This pool allows components of the server to reuse threads to eliminate the need to create new threads at runtime. Creating new threads is typically a time and resource-intensive operation. This window gives you the ability to set up the thread pool by assigning a range of threads the pool will be able to create, when the pool should time out inactive threads, and whether or not you want the pool to create new threads if it has more requests than the maximum size allows.
- **Custom properties:** Here you can supply custom properties that you would like to be accessible programmatically to your messaging beans. By supplying the properties here, the context object in the JVM will allow you beans operating under this context to access these properties through access methods.

► **ORB service properties:** This page allows you to specify settings for the Object Request Broker service. The Configuration tab lists the general settings you will want to configure for the ORB service running on WebSphere. The following are the attributes that you can select:

- **Request timeout:** Used to specify the number of seconds you want the ORB service to wait before timing out a request message. This value defaults to 180.
- **Request retries count:** Specifies the number of times that the ORB attempts to send a request if the server fails. This ability to retry a request can enable application recovery from transient network failures. This value defaults to 1.
- **Request retries delay:** This value specifies the number of milliseconds to wait between retries. This value defaults to 0.
- **Connection cache maximum:** Used with the connection cache minimum to set a range that the ORB service can use to keep a certain number of connections available in the cache. The maximum is the largest number of connections allowed to occupy the connection cache for the service. This value defaults to 240.
- **Connection cache minimum:** This attribute specifies the number of connections allowed to occupy the connection cache. The minimum value defaults to 100.
- **ORB Tracing:** Used to enable the trace service at the ORB level. This will provide trace information on GIOP messages generated by the ORB service. Many ORB exceptions may be difficult to isolate. This tracing

option will help you isolate those situations where you have received a GIOP message and need to troubleshoot the problem.

This setting affects two system properties, com.ibm.CORBA.Debug, and com.ibm.CORBA.CommTrace. By default this option is not enabled. If you choose to enable it you will have to ensure that the above system properties have been set to `true`. If they are not both set to `true`, you will not be able to trace the GIOP messages.

– **Locate request timeout:** This value specifies the number of seconds that a LocateRequest message will wait for a response.

  The LocateRequest message enables clients to determine whether an object reference is known, whether the server in question can process a request to this object, and if not, the server to which requests for this object are made. The information obtained by LocateRequest is also provided by Request, but in the case of a Location Forward situation, it enables clients to avoid a potentially lengthy transmission of parameters associated with a particular request.

– **Force-tunnel:** This option controls how the client ORB attempts to use HTTP tunneling. The values of this optional property can be:

  - **Always:** Use HTTP tunneling immediately, without trying TCP connections first.
  - **Never**: Disable HTTP tunneling. If a connection fails, a CORBA system exception will be thrown (COMM_FAILURE).
  - **When Required:** Use HTTP tunneling if the TCP connections fail.

– **Tunnel agent URL:** This must be specified if the force-tunnel option is chosen to use HTTP tunneling. The URL must be properly formatted for the tunneling to work. An example of the proper format is:

```
http://w3.mycorp.com:81/servlet/com.ibm.CORBA.
services.IIOPTunnelServlet
```

  or, for applets:

```
http://applethost:port/servlet/com.ibm.CORBA.services. IIOPTunnelServlet
```

– **Pass by reference:** When parameters are passed locally, they are passed “by value” using the standard IIOP copy semantics. Use this property to select “pass by reference” instead. Depending on your application design, using pass by reference might introduce side effects, such as unexpected or unpredictable behavior. Use this option with caution.

– **Thread pools:** For information on thread pools and determining their size, see *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.

- **Administration services:** This page allows you to specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts. These settings are not something you would normally be concerned with. If you plan to extend the administration services by adding custom MBeans, see the “Extending WebSphere Application Server Administrative System with custom MBeans“ topic in the InfoCenter.
- **Debugging Service:** This page allows you to specify settings for the debugging service, to be used in conjunction with a workspace debugging client application, for example, the Application Server Toolkit or WebSphere Studio.
- **Custom Services:** This page allows you to define custom service classes that will run within this server, and their configuration properties.
- **Process Definition:** A process definition defines the command-line information necessary to start/initialize a process. The process definition properties page contains the following:
  - **Executable name:** This defaults to the JVM packaged with WebSphere Application Server.
  - **Executable arguments:** This is a set of additional arguments to be passed to the JVM.
  - **Working directory:** This specifies the relative root for searching files. For example, if you do a File.open(“foo.gif”), foo.gif must be present in the working directory to be found. This directory will be created by WebSphere if it does not exist. We recommend that you create a specific working directory for each application server, for example /projects/webbank. The default working directory is ${USER_INSTALL_ROOT}. As for any path, we recommend that you create a variable for the working directory, rather than hardcoding it.

    Note: The GA version (that is, with the FixPak applied) of the product won’t create the working directory if you use a composed path, such as F:/WebbankSample/workingDir. If you wish to use such a path, make sure to create it before starting the application server, or the startup sequence will fail.

  - **Java virtual machine definition**, such as the initial and maximum heap sizes, debug options, the process classpath, or different runtime options such as profiler support and heap size.
  - **Process execution settings**, such as the process priority, or the user/group that should be used to run the process.

  - **Environment Entries:** a set of name/value pairs (such as LD_LIBRARY_PATH).
  - **Monitoring policy**: These properties determine how the node agent will monitor the application server. It includes ping intervals and timeouts.

- **Performance Monitoring Service:** This page allows you to specify settings for performance monitoring. These settings are discussed in detail in *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.
- **End Points:** This page contains the basic port definitions for the server. Endpoint definitions consist of a host name and port pair. The following ports are defined with these settings:
  - BOOTSTRAP_ADDRESS: Used by the naming client to specify the naming server to look up the initial context. Every WebSphere Application Server V5 process, except the JMS server, has a bootstrap server.
  - SOAP_CONNECTOR_ADDRESS: The port that is used by HTTP transport for incoming SOAP requests. JMX management uses the SOAP connector address. port.
  - DRS_CLIENT_ADDRESS: Used to configure the Data Replication Service (DRS) which is a JMS-based message broker system for dynamic caching.

  The next three ports are related to security. For more information about security, see *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

  - CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS
  - CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS
  - SAS_SSL_SERVERAUTH_LISTENER_ADDRESS

  You may not ever need to manually change these ports. It is likely, however, that you will want to view these. For example, if you use the dumpNameSpace command, you can specify the bootstrap port of the process to dump the name space from.

- **Classloader:** Classloaders are discussed in 14.6, “Understanding WebSphere classloaders” on page 687.

Starting an application server

Before managing a server in a Network Deployment environment using the administrative console, you must make sure that the node agent for the server’s node is running. To do this:

1. Select **System Administration -> Node Agents**.

2. The status of the node agent is in the far right column. If it hasn't started, you must start it from the command line of the node using the following command:

```
<WAS_HOME>/bin/startnode (.sh)
```

Viewing server status

To check the status of a server, follow these steps:

1. **Servers -> Application Servers**.
2. The servers will be listed. The last column to the right contains an icon to indicate the status of each server. Figure 8-21 shows the icons and the status that they indicate.

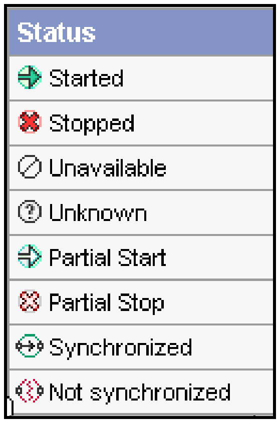


Figure 8-21 Status icons

Starting a server

An application server can be started using one of the following options.

Method 1

From the administrative console:

1. Select **Servers -> Application Servers**.
2. Check the box(s) to the left of the server(s).
3. Click **Start**.

Method 2

From the command line:

1. Change directory to the bin directory of the base application server installation for that node.
2. Run **`startServer <appserver name>`**.

3. If there are any errors, check the log file for the application server process:

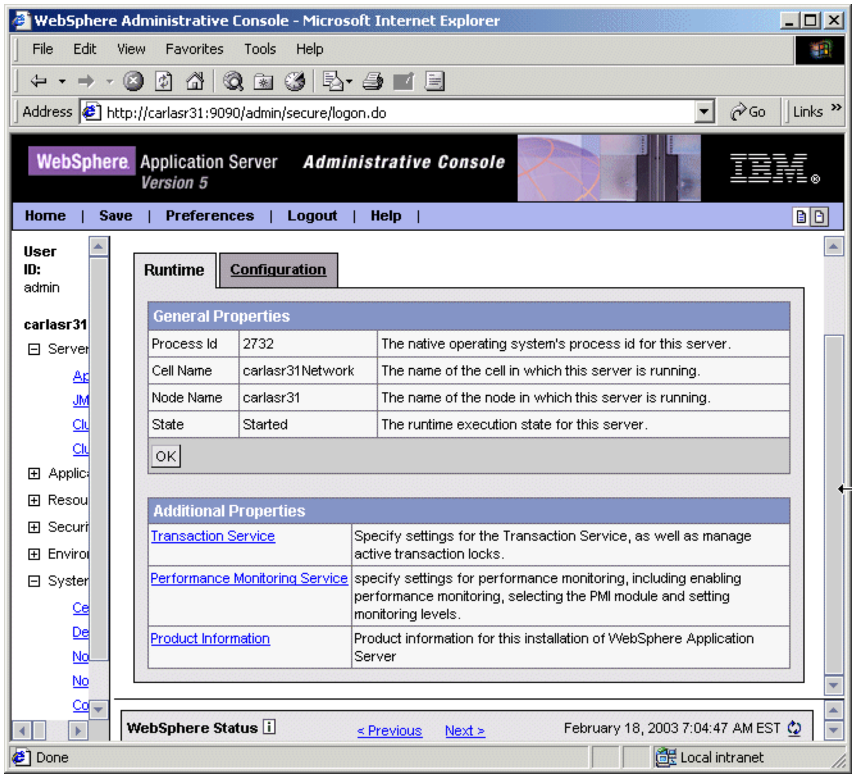
   <WAS_HOME>/logs/<server_name>/SystemOut.log

Note: In version 5.0, you had the option of setting an initial state for an application server. This option has been removed in version 5.1.

By default, all the applications on a server start when the application server starts. To prevent an application from starting, see “Preventing an enterprise application from starting on a server” on page 318.

Viewing runtime attributes

1. Select **Servers -> Application Servers**.
2. The servers will be listed. Click a server.
3. If the server is running, you will see both a Configuration tab and Runtime tab. If it isn’t running, you will only see a Configuration tab.
4. Click the **Runtime** tab. Figure 8-22 on page 300 shows the Runtime tab and the information it provides.



WebSphere Administrative Console - Microsoft Internet Explorer

Address http://carlasr31:9090/admin/secure/logon.do

User ID: admin

carlasr31

Runtime | Configuration

General Properties

Process Id	2732	The native operating system's process id for this server.
Cell Name	carlasr31Network	The name of the cell in which this server is running.
Node Name	carlasr31	The name of the node in which this server is running.
State	Started	The runtime execution state for this server.

Additional Properties

Transaction Service	Specify settings for the Transaction Service, as well as manage active transaction locks.
Performance Monitoring Service	specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.
Product Information	Product information for this installation of WebSphere Application Server

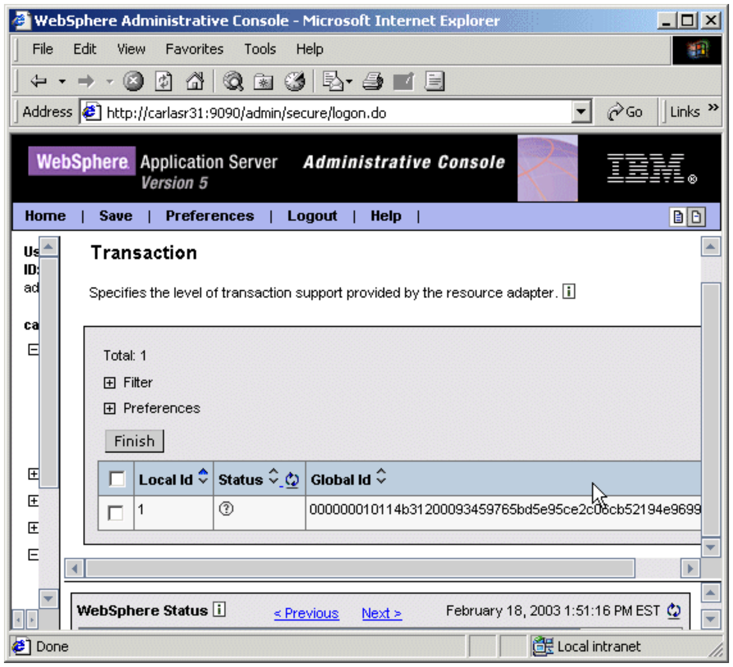
WebSphere Status < Previous Next > February 18, 2003 7:04:47 AM EST

Figure 8-22 Application server Runtime tab

5. In the Additional Properties table, you have access to the following:

   – **Transaction Service** settings. These are the same settings discussed in “Transaction service properties: The transaction service properties allow you to specify settings for the transaction service, as well as manage active transaction locks.” on page 288. You can change the timeout settings while the server is running, but not the transaction log directory setting.

     The Manage Transaction button allows you to view current transaction activity in the server. If there is no transaction activity, the window will not change and a message will appear in the message box at the top indicating there are no current transactions. If there are transactions running you will see a window like Figure 8-23 on page 301.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |

Transaction

Specifies the level of transaction support provided by the resource adapter.

Total: 1

⊞ Filter

⊞ Preferences

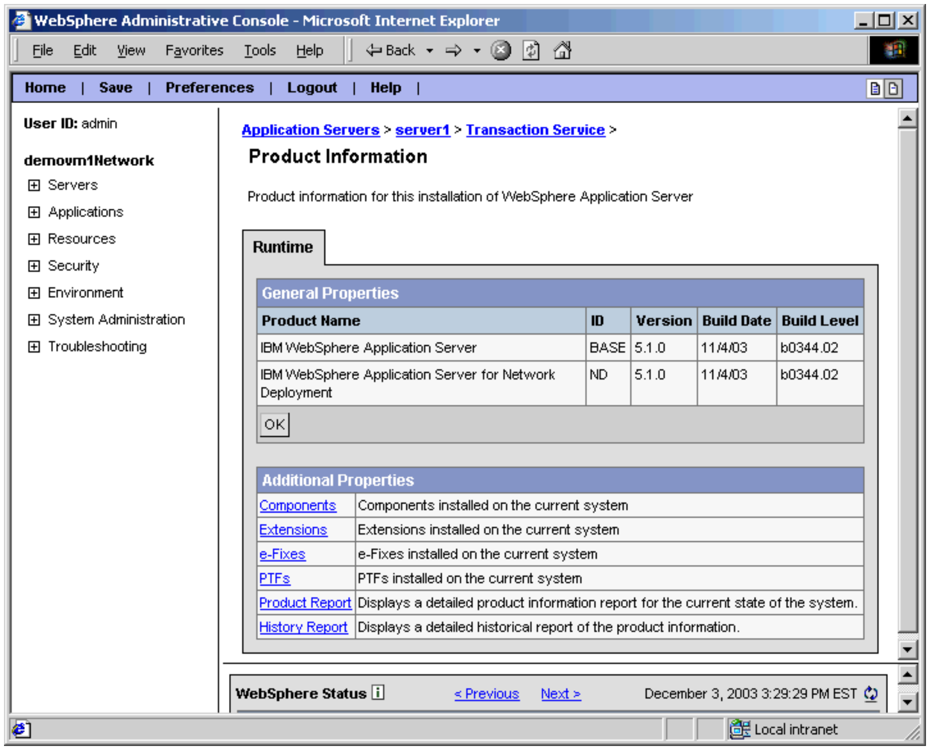
Finish

	Local Id	Status	Global Id
☐	1	ⓘ	000000010114b31200093459765bd5e95ce2c06cb52194e9699

WebSphere Status
< Previous
Next >
February 18, 2003 1:51:16 PM EST
Done
Local intranet

Figure 8-23 Using the Manage Transactions button

- **Performance Monitoring Service** settings. This allows you to change the instrumentation levels while the server is running. These settings are discussed in detail in *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.
- **Product Information.** This option gives you access to extensive information about the product installation and FixPak information.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Home | Save | Preferences | Logout | Help |
User ID: admin
demovm1Network
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
Application Servers > server1 > Transaction Service >
Runtime
WebSphere Status
< Previous
Next >
Local intranet

Product Information

Product information for this installation of WebSphere Application Server

General Properties

Product Name	ID	Version	Build Date	Build Level
IBM WebSphere Application Server	BASE	5.1.0	11/4/03	b0344.02
IBM WebSphere Application Server for Network Deployment	ND	5.1.0	11/4/03	b0344.02

OK

Additional Properties

Components	Components installed on the current system
Extensions	Extensions installed on the current system
e-Fixes	e-Fixes installed on the current system
PTFs	PTFs installed on the current system
Product Report	Displays a detailed product information report for the current state of the system.
History Report	Displays a detailed historical report of the product information.

December 3, 2003 3:29:29 PM EST

Figure 8-24   Product information

Stopping an application server

An application server can be stopped using one of the following options.

Method 1

From the administrative console:

1. Select **Servers -> Application Servers**.
2. Check the box(s) to the left of the server(s).
3. Click **Stop**.

Method 2

From the command line:

1. Change directory to the bin directory of the base application server installation for that node.

2. Run **`stopServer <appserver name>`**. For example:

   ```
   cd c:\websphere\appserver\bin
   stopserver server1
   ```

If there are any errors, check the log file for the application server process:

   <WAS_HOME>/logs/<server_name>/SystemOut.log

> **Note:** If you stop the application server, all applications served by that application server will become unavailable.

Method 3

Stopping a node also stops the node agent and all servers on the node.

1. From the administrative console, select **System Administration -> Nodes**.
2. Check the box beside the node and click **Stop**.

Method 4 (Restarting the servers on a node)

Stopping a node also stopsthe node agent and all servers on the node.

1. From the administrative console, select **System Administration -> Node Agents**.
2. Check the box for the server.
3. Click **Restart all Servers on the Node**.

8.6.4 Managing clusters

This section discusses creating, configuring, and managing clusters using the administrative console. Clustering is not an option in a base installation. Before designing a WebSphere installation using clustering, you should review *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.

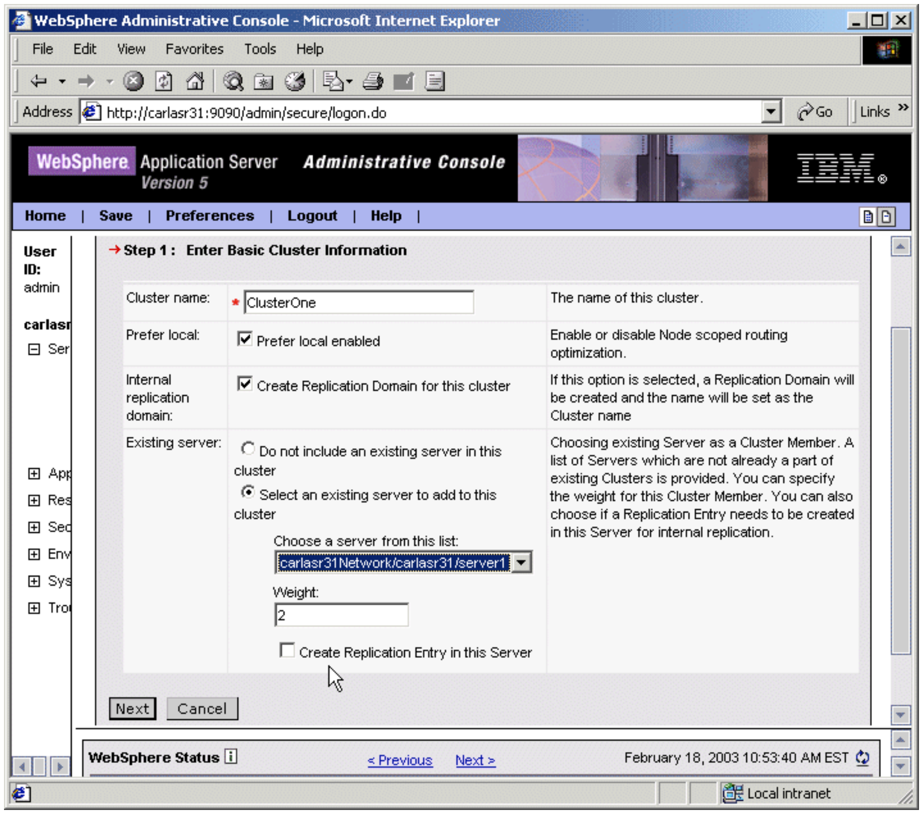
Creating clusters

Cluster members are created either during cluster creation, or by specifying an existing application server to become a member of a cluster. When creating a cluster, it is possible to select the template of an existing application server for the cluster without adding that application server into the new cluster. For this reason, consider creating a server with the server properties that you want as a standard in the cluster first, then use that server as a template or as the first server in the cluster.

To create a new cluster:

1. Select **Servers -> Clusters**.

2. Click **New**.



WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin

→ Step 1 : Enter Basic Cluster Information

Cluster name:	* ClusterOne	The name of this cluster.
Prefer local:	☑ Prefer local enabled	Enable or disable Node scoped routing optimization.
Internal replication domain:	☑ Create Replication Domain for this cluster	If this option is selected, a Replication Domain will be created and the name will be set as the Cluster name
Existing server:	○ Do not include an existing server in this cluster ● Select an existing server to add to this cluster Choose a server from this list: carlasr31Network/carlasr31/server1 Weight: 2 ☐ Create Replication Entry in this Server	Choosing existing Server as a Cluster Member. A list of Servers which are not already a part of existing Clusters is provided. You can specify the weight for this Cluster Member. You can also choose if a Replication Entry needs to be created in this Server for internal replication.

Next Cancel
WebSphere Status < Previous Next > February 18, 2003 10:53:40 AM EST
Local intranet

Figure 8-25 Creating a new cluster

3. Enter the information for the new cluster:

   – **Cluster name:** Enter a cluster name of your choice.

   – **Prefer local:** This setting indicates that a request to an EJB should be routed to an EJB on the local node if available.

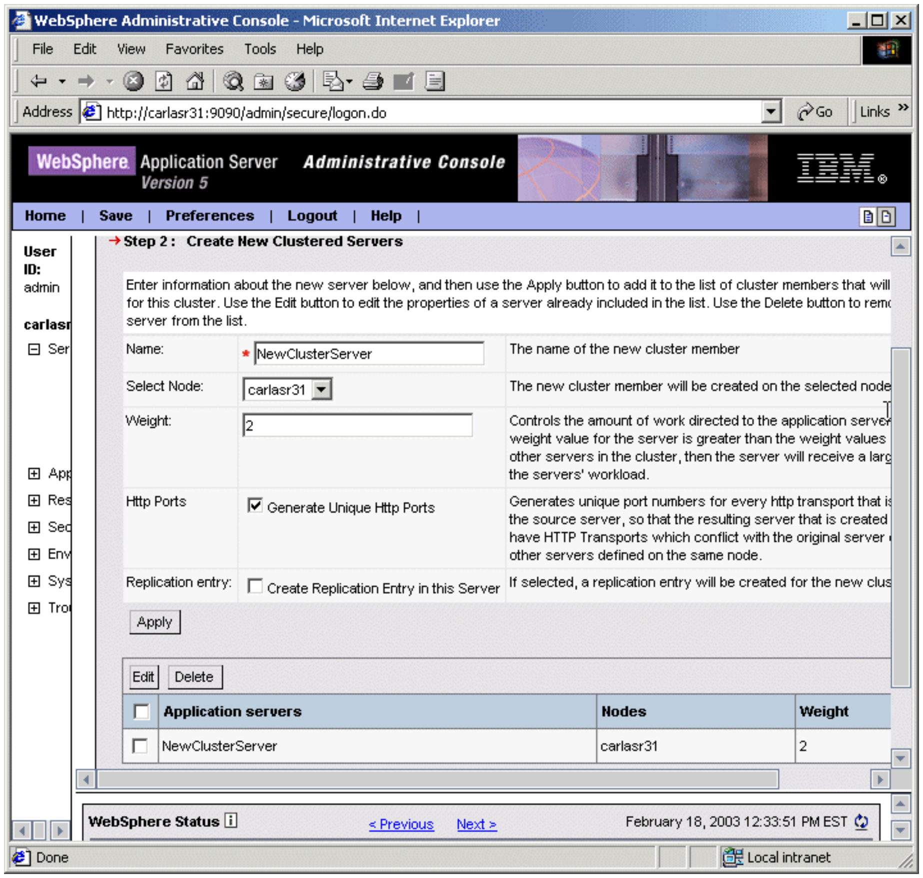
   – **Internal replication domain:** Indicates you want to use memory-to-memory replication for persistent session management and that a replication domain should be created. Replication domains are discussed in 9.9.2, “WebSphere internal messaging” on page 361.

– **Existing server:** One application server must be added to the cluster. You can choose an existing application server from the list and/or create new application servers for the cluster.

  If you want to use an existing server, select **Select an existing server to add to this cluster** and click **Next**. The server you select will be added to the cluster and you will be given the opportunity to add new servers also.

  If you will not use an existing server, select **Do not include an existing server** in this cluster and click **Next**.

4. The next window allows you to add servers to the cluster. For each server, enter a name, select the appropriate parameters, and click **Apply**. The servers in the cluster will be listed at the bottom of the window.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr
Ser
App
Res
Sec
Env
Sys
Tro
Done
Local intranet

→Step 2 : Create New Clustered Servers

Enter information about the new server below, and then use the Apply button to add it to the list of cluster members that will for this cluster. Use the Edit button to edit the properties of a server already included in the list. Use the Delete button to rem server from the list.

Name:	* NewClusterServer	The name of the new cluster member
Select Node:	carlasr31	The new cluster member will be created on the selected node
Weight:	2	Controls the amount of work directed to the application serve weight value for the server is greater than the weight values other servers in the cluster, then the server will receive a larg the servers' workload.
Http Ports	☑ Generate Unique Http Ports	Generates unique port numbers for every http transport that i the source server, so that the resulting server that is created have HTTP Transports which conflict with the original server other servers defined on the same node.
Replication entry:	☐ Create Replication Entry in this Server	If selected, a replication entry will be created for the new clus

Apply

Edit Delete

☐	Application servers	Nodes	Weight
☐	NewClusterServer	carlasr31	2

WebSphere Status < Previous Next > February 18, 2003 12:33:51 PM EST

Figure 8-26 Creating a new cluster

– **Name:** A name for the new server.

  – **Server weight:** Determines how workload is distributed. For example, If all cluster members have identical weights, work will be distributed among the cluster members equally. Servers with higher weight values are given more work. A rule of thumb formula for determining routing preference would be:

    ```
    % routed to Server1 = weight1 /(weight1+weight2+...+weight n)
    ```

    where there are `n` cluster members in the cluster.

  – **Replication entry:** For information about replication domains and replication entries, see 9.9.2, “WebSphere internal messaging” on page 361.

5. When all the servers have been entered, click **Next**.
6. A summary page will be presented to show you what will be created.
7. Click **Finish** to create the cluster and new servers.
8. Save the configuration.

Viewing cluster topology

The administrative console provides a graphical view of the existing clusters and their members. To see the view:

1. Select **Servers -> Cluster Topology**.
2. Expand each category.

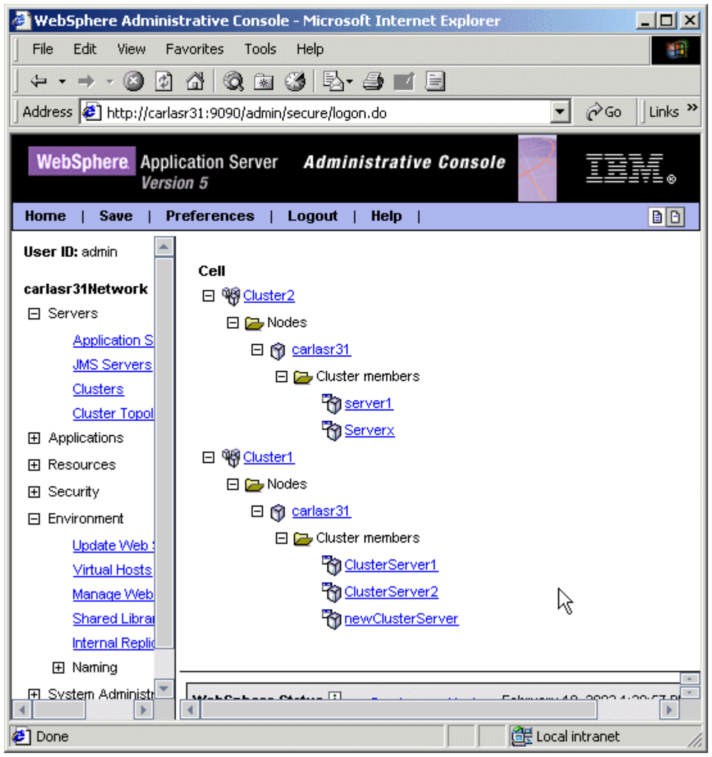
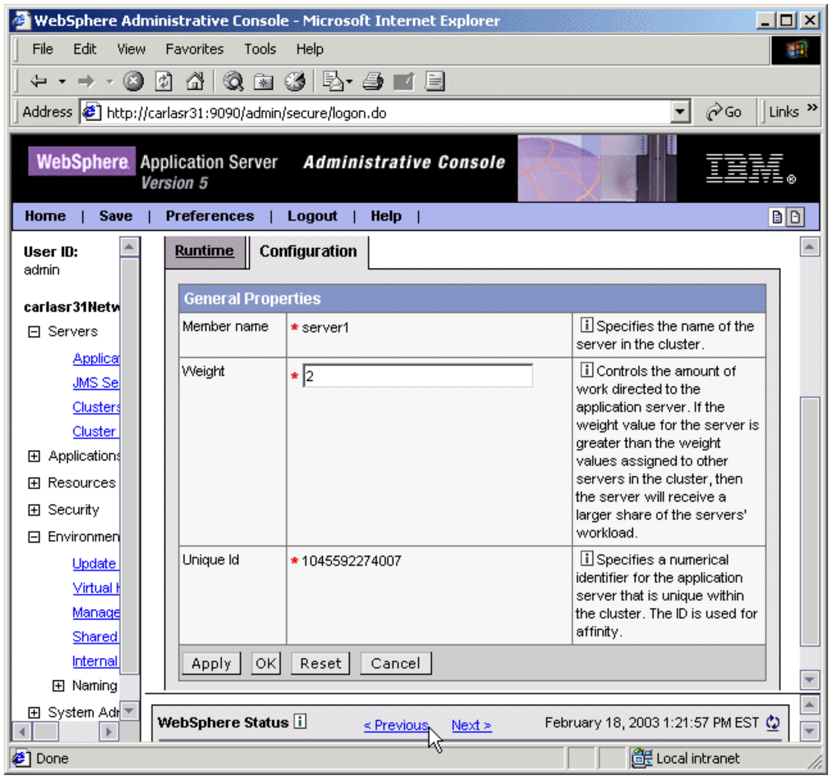


Figure 8-27 Cluster topology view

3. Selecting a server will display the attributes of the server as related to the cluster.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Netw
Servers
Applica
JMS Se
Clusters
Cluster
Applications
Resources
Security
Environmen
Update
Virtual H
Manage
Shared
Internal
Naming
System Adr
Runtime
Configuration
Apply
OK
Reset
Cancel
WebSphere Status
< Previous
Next >
February 18, 2003 1:21:57 PM EST
Done
Local intranet

General Properties

Member name	* server1	Specifies the name of the server in the cluster.
Weight	* 2	Controls the amount of work directed to the application server. If the weight value for the server is greater than the weight values assigned to other servers in the cluster, then the server will receive a larger share of the servers' workload.
Unique Id	* 1045592274007	Specifies a numerical identifier for the application server that is unique within the cluster. The ID is used for affinity.

Figure 8-28 Viewing the cluster characteristics of a server

Managing clusters

Application servers within a cluster can be managed as independent servers. A second option is to manage all the servers in the cluster using a single button:

1. Select **Servers -> Clusters**.
2. Check each cluster you want to work with and select one of the following options:
   - **Start:** starts all servers in the cluster.
   - **Stop:** Stops all servers in the cluster. This allows the server to finish existing requests and allows failover to another member of the cluster.
   - **Ripplestart:** Stops, then starts all servers in the cluster.
   - **ImmediateStop:** Stop all servers immediately.

8.6.5 Managing virtual hosts

Understanding virtual hosts: This section tells you how to create or modify a virtual host using the administrative console. For additional information about virtual hosts, see the following:

- For information on what a virtual host is and how it is used, see 2.7, “Virtual hosts” on page 51.
- For an example of defining and using a new virtual host, see 14.1.4, “Defining the Webbank virtual host” on page 654.
- For information on creating and modifying virtual host definitions with wsadmin, see “Creating a virtual host” on page 814 and “Modifying a virtual host” on page 815.

A virtual host is a configuration enabling a single host machine to resemble multiple host machines. It consists of host alias(es), which consist of a host name and a port number. If “*” is specified as a host name, all host names and IP addresses that the Web server can receive will be mapped to that virtual host.

There are two virtual hosts defined during installation, default_host and admin_host.

- The default_host virtual host is intended for access to user applications, either through the HTTP transport or via a Web server. At installation time, it is configured as the default virtual host for the server1 application server. It is configured to match requests to ports 80, 9080, and 9443 for any host name.
- The admin_host virtual host is used for access to the WebSphere administrative console. It is configured to match requests to the secure ports 9090 (HTTP transport) and 9043 (Web server) for any host name.

When you install an application, you associate a virtual host with each Web module in the application. By associating a virtual host with a Web module, requests that match the host aliases for the virtual host should be processed by servlets/JSPs in this Web module. The Web server plug-in also checks the URI of the request against the URIs for the Web module to determine whether the Web module can handle them or not.

A single virtual host can be associated with multiple Web modules unless each application has unique URIs. If there are same URIs among applications, different virtual hosts must be created and associated with each of the applications.

Creating a virtual host

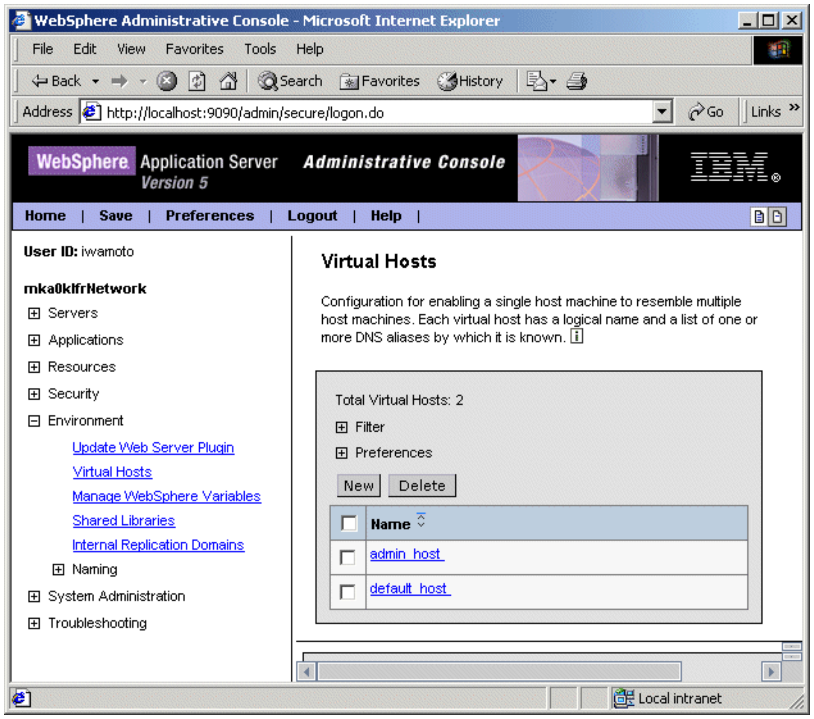
By default, default_host is associated with all user application requests. There are some cases that multiple virtual hosts should be created, for example:

- Applications having conflicting URIs
- To support extra ports such as port 443 for SSL
- To keep clear independence of each virtual host for applications/servers

After a virtual host is created, the Web server plug-in needs to be updated. For information on how to do this, see "Generating the Web server plug-in" on page 330.

The configuration of a virtual host is applied to an entire cell. To create a new virtual host:

1. Select **Environment -> Virtual Hosts** and then click **New**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Troubleshooting
Local intranet

Virtual Hosts

Configuration for enabling a single host machine to resemble multiple host machines. Each virtual host has a logical name and a list of one or more DNS aliases by which it is known.

Total Virtual Hosts: 2

Filter

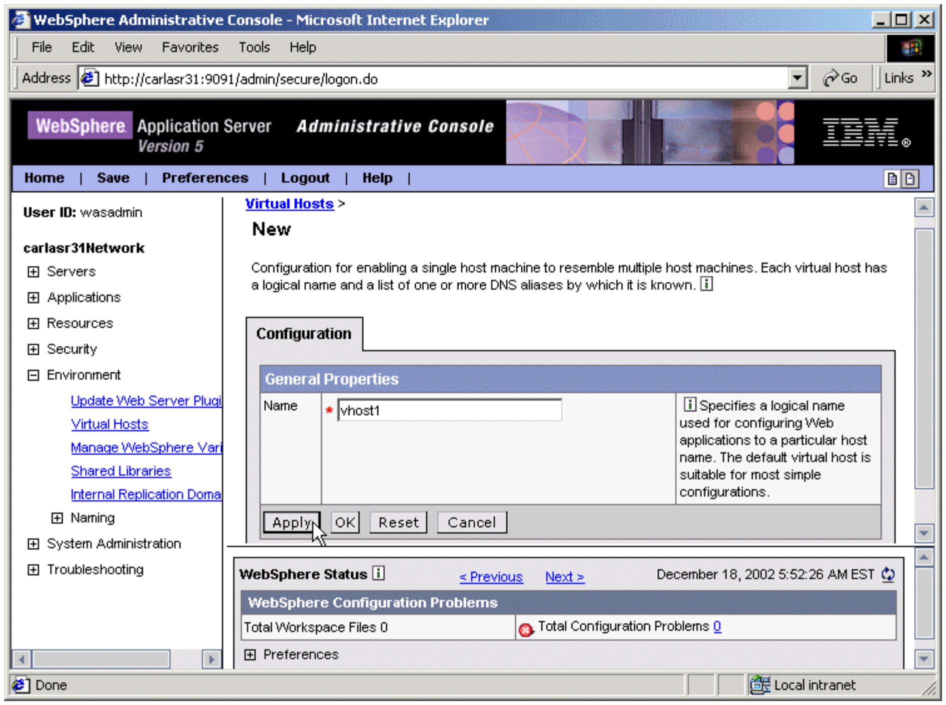
Preferences

New Delete

	Name
☐	admin_host
☐	default_host

Figure 8-29 Creating a virtual host

2. Enter a name for the virtual host and click **Apply**.


WebSphere Administrative Console - Microsoft Internet Explorer

Address http://carlasr31:9091/admin/secure/logon.do

User ID: wasadmin

carlasr31Network

Virtual Hosts >

New

Configuration for enabling a single host machine to resemble multiple host machines. Each virtual host has a logical name and a list of one or more DNS aliases by which it is known.

Configuration

General Properties

Name	* vhost1	Specifies a logical name used for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

WebSphere Status

December 18, 2002 5:52:26 AM EST

WebSphere Configuration Problems

Total Workspace Files 0	Total Configuration Problems 0

Figure 8-30 Virtual host configuration

3. Click **Host Aliases** in the Additional Properties pane.
4. Click **New**.

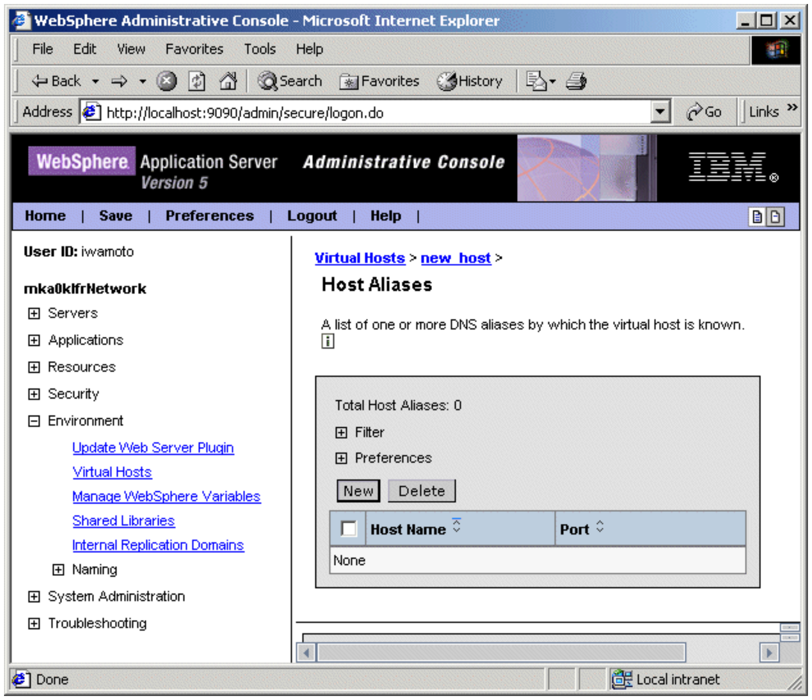

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Troubleshooting
Virtual Hosts > new_host >
Host Aliases
A list of one or more DNS aliases by which the virtual host is known.
Total Host Aliases: 0
Filter
Preferences
New
Delete
Host Name
Port
None
Done
Local intranet

Figure 8-31 Host aliases

5. Enter values for the Host Name and Port fields and click **OK**.

   The host aliases are not necessarily the same as the host name and port number of the WebSphere Application Server(s). They are the host name(s) and port number(s) that the Web server plug-in is expecting to receive from the browser. The Web server plug-in will send the request to the application server using the host name and port number in the transport setting for that server. If the Web server is running on a separate machine from WebSphere, then the host aliases are for Web server machines.

   Mapping HTTP requests to host aliases is case sensitive and the match must be alphabetically exact. Also, different port numbers are treated as different aliases.

   For example, the request:

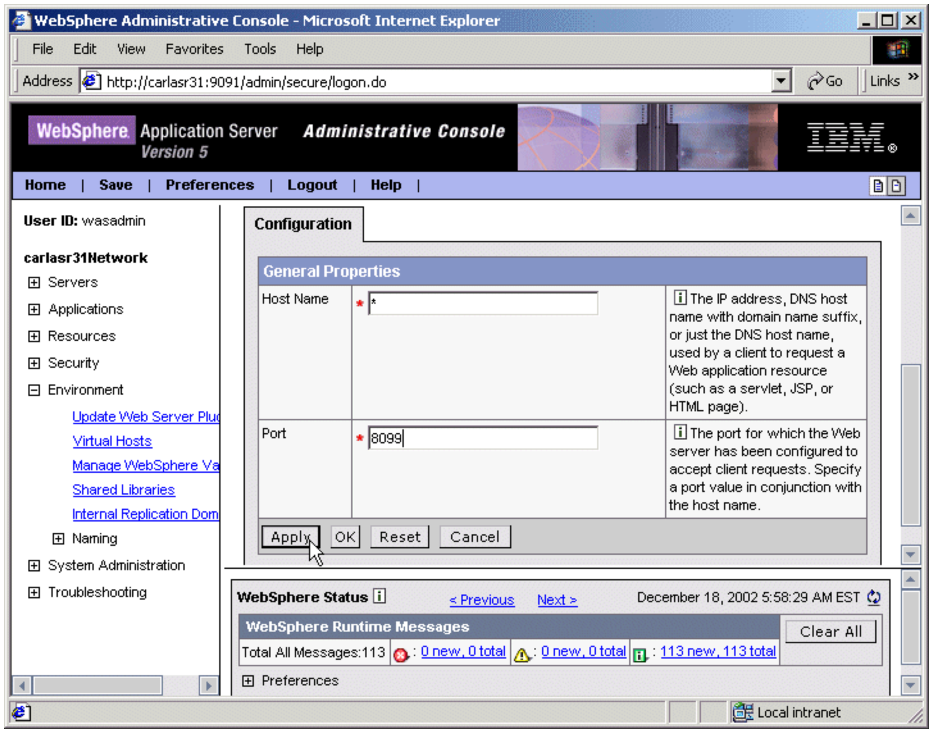
   – http://www.myhost.com/myservlet

   does *not* map to any of the following:

   – http://myhost/myservlet
   – http://www.myhost.com/MyServlet
   – http://www.myhost.com:9876/myservlet

If the Web server plug-in receives a request that does not match one of the virtual hosts, an HTTP error will be returned to the user.

Simple wild cards can be used in the host aliases. A “*” may be used for the host name or the port or both. It means that any request will match this rule.



Host Name	*	The IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP, or HTML page).
Port	8099	The port for which the Web server has been configured to accept client requests. Specify a port value in conjunction with the host name.

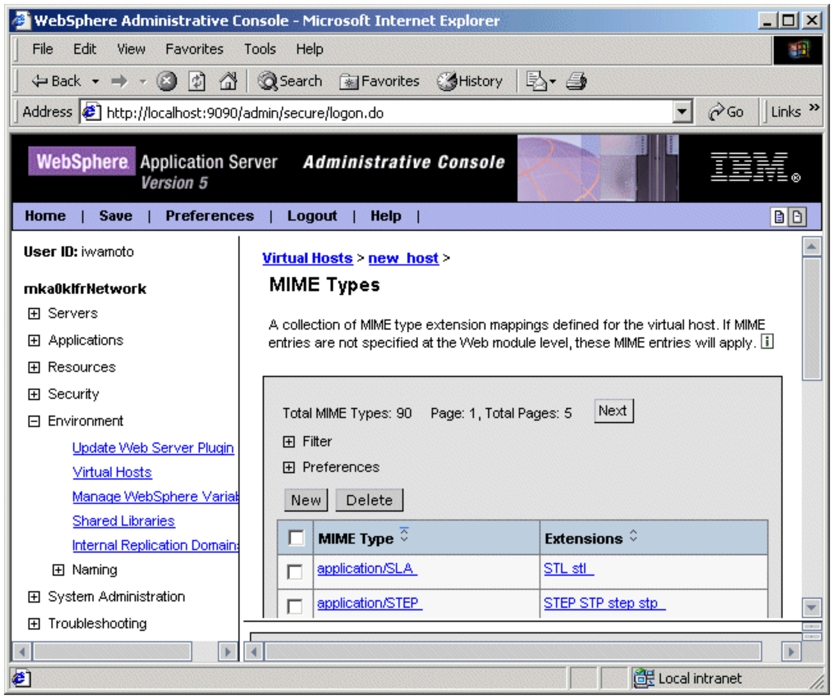
December 18, 2002 5:58:29 AM EST

Total All Messages:113 : 0 new, 0 total : 0 new, 0 total : 113 new, 113 total

Figure 8-32   New host alias

Note: If the virtual host is used in a cluster environment, all host aliases used by servers in the cluster should be registered in the virtual host.

6. Multi-Purpose Internet Mail Extensions (MIME) mappings associate a file name extension with a type of data file (text, audio, image). A set of MIME types is automatically defined for you when you create a virtual host. To see or alter the MIME types associated with this new virtual host, click **MIME Types** in the Additional Properties pane of the virtual host.
7. Click **New** to add a MIME type.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: iwamoto
mka0klfrNetwork
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variab
Shared Libraries
Internal Replication Domain
Naming
System Administration
Troubleshooting
Local intranet

Virtual Hosts > new_host >

MIME Types

A collection of MIME type extension mappings defined for the virtual host. If MIME entries are not specified at the Web module level, these MIME entries will apply.

Total MIME Types: 90 Page: 1, Total Pages: 5 Next

Filter

Preferences

New Delete

	MIME Type	Extensions
☐	application/SLA	STL stl
☐	application/STEP	STEP STP step stp

Figure 8-33 MIME types

8. Enter the MIME type and extension. Click **Apply** to continue adding new types or click **OK** if done.
9. Click **Save** on the taskbar and save your changes.

Important: If you create, delete, or update virtual hosts, you need to regenerate the Web server plug-in.

8.6.6 Managing enterprise applications

This section describes how to perform common administration tasks on enterprise applications using the administrative console.

Installing an enterprise application

To install an enterprise application into a WebSphere configuration, you must install its modules onto one or more application servers. The steps for this task are as follows:

1. Select **Applications -> Enterprise Applications -> Install**, or **Applications -> Install New Application.**

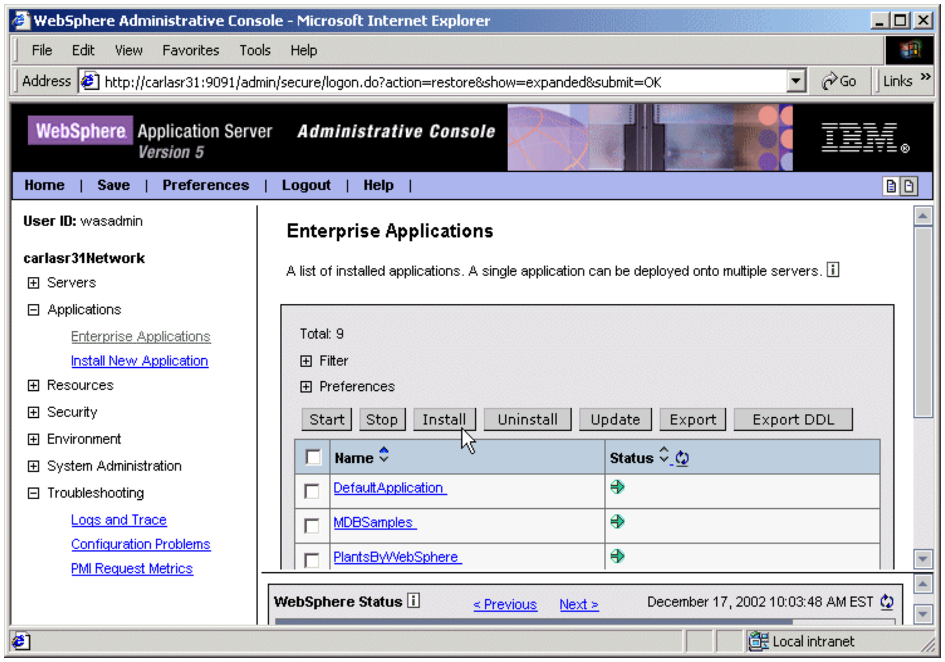
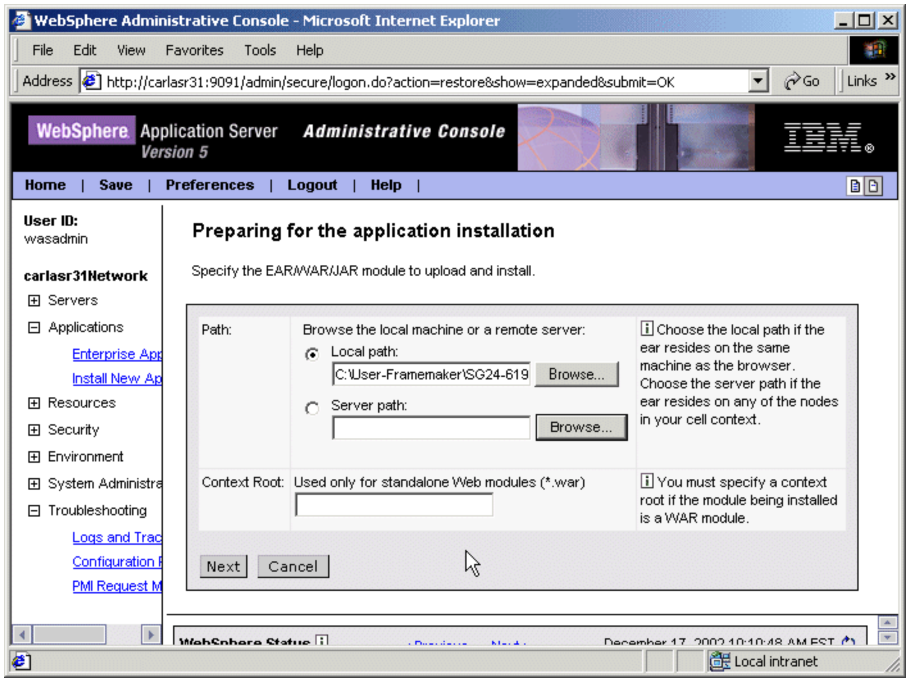

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Applications
Install New Application
Resources
Security
Environment
System Administration
Troubleshooting
Logs and Trace
Configuration Problems
PMI Request Metrics
Enterprise Applications
A list of installed applications. A single application can be deployed onto multiple servers.
Total: 9
Filter
Preferences
Start Stop Install Uninstall Update Export Export DDL
Name Status
DefaultApplication
MDBSamples
PlantsByWebSphere
WebSphere Status < Previous Next > December 17, 2002 10:03:48 AM EST
Local intranet

Figure 8-34 Working with enterprise applications

2. Specify the location of the EAR file to install, as shown in Figure 8-35 on page 316.

   The EAR file that you are installing can be either on the client machine (the machine that runs the Web browser) or on any of the nodes in the cell. Click **Next**.


Preparing for the application installation

Specify the EAR/WAR/JAR module to upload and install.

Path: Browse the local machine or a remote server:
[✓ Local path:] C:\User-Framemaker\SG24-619
Server path:

Choose the local path if the ear resides on the same machine as the browser. Choose the server path if the ear resides on any of the nodes in your cell context.

Context Root: Used only for standalone Web modules (*.war)

You must specify a context root if the module being installed is a WAR module.

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise App
Install New Ap
Resources
Security
Environment
System Administra
Troubleshooting
Logs and Trac
Configuration
PMI Request M
Browse...
Next
Cancel
Local intranet

Figure 8-35 Installing an enterprise application

3. The next window has settings that will be using during the installation. The settings have to do with bindings and the default host. Click **Next**.
4. The rest of the installation process is done in the following ten steps:
   a. Step 1: Provide options to perform the installation.
   b. Step 2: Provide JNDI names for beans.
   c. Step 3: Provide default data source mapping for modules containing EJB 2.0 entity beans.
   d. Step 4: Map data sources for all CMP 2.0 data beans.
   e. Step 5: Map EJB references to beans.
   f. Step 6: Map virtual hosts for Web modules.
   g. Step 7: Map modules to application servers.
   h. Step 8: Map security roles to users/groups.
   i. Step 9: Ensure all unprotected 1.0 methods have the correct level of protection.
   j. Step 10: Summary.

5. Click **Finish** to install the application.
6. Save the configuration.

Note: When you install an application in a Network Deployment environment, the binaries are stored on the application server node. The configuration files are stored on the Network Deployment node and on the application server node. The default locations are:

Binaries:

- <WAS_HOME>/installedApps/<cell_name>/applications

Configuration:

- <WAS_HOME>/config/cells/<cellname>/applications/<appname>.ear
- <WAS_ND_HOME>/config/cells/<cellname>/applications/<appname>.ear

Upgrading an enterprise application

To upgrade an enterprise application, perform the following steps:

1. Select **Applications -> Enterprise Applications**.
2. Check the application you want to upgrade and then click **Upgrade**.
3. Specify the location of the Enterprise Application Archive (EAR) containing the new version of the application.
4. The remainder of the steps required are the same as those described for installation of a new enterprise application.

Note: In some cases, it may be more appropriate to manually edit the installed application to update its files, rather than using the full upgrade approach.

Uninstalling an enterprise application

To uninstall an enterprise application that is no longer needed, do the following:

1. Select **Applications -> Enterprise Applications**.
2. Check the application you want uninstall and click **Uninstall**.

Exporting an enterprise application

If you have modified the binding information of an enterprise application, you may want to export the changed bindings to a new EAR file. To export an enterprise application to an EAR file:

1. Select **Applications -> Enterprise Applications**.
2. Check the application you want to export and click **Export**.

3. Click the link for the file you want to export.
4. Specify the directory on the local machine where the export is to be saved and click **OK**.

Starting an enterprise application

An application can be started by following these steps. From the administrative console:

1. Select **Applications -> Enterprise Applications**.
2. Check the application you want started and click **Start**.

Note: In order to start an application, the application server that contains the application has to be started. If not, the application will show in the administrative console as unavailable and you won’t be able to start it.

There is no command-line option to start an application, and there is no Initial State setting either. The application will always be started when its application server starts.

Stopping an enterprise application

An application can be stopped using one of these options:

1. From the administrative console.
   a. Select **Applications -> Enterprise Applications**
   b. Check the application you want to stop and click **Stop**.

Preventing an enterprise application from starting on a server

By default, an application will start when the server starts. The only way to prevent this is to disable the application from running on the server.

1. From the administrative console:
   a. Select **Applications -> Enterprise Applications**
   b. Click the application to open the configuration.
   c. Select **Target Mappings** in the Additional Properties table.
   d. Select the server for which you want to disable the application.
   e. Deselect the Enable option and click **OK**.
   f. Save the configuration.

Viewing installed applications

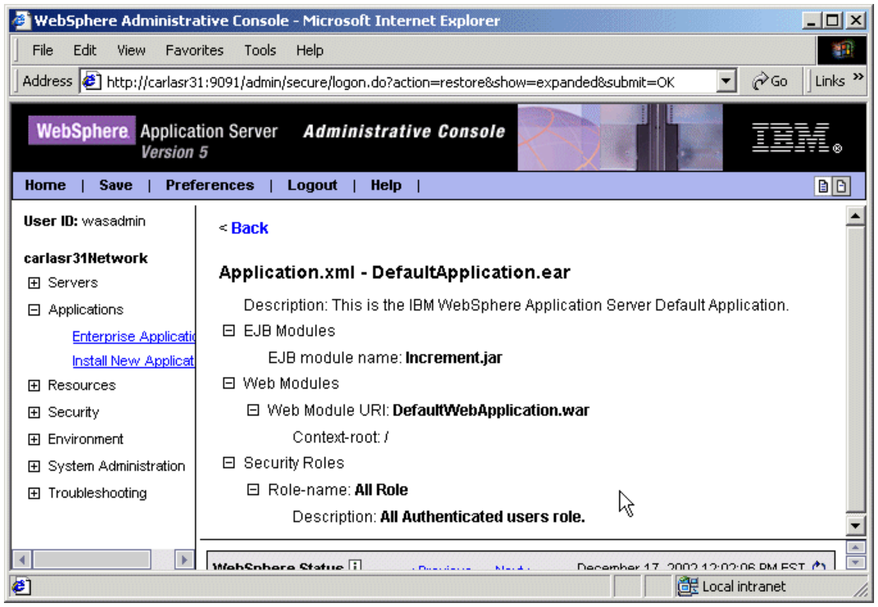
The administrative console does not display the deployed servlets, JSPs or EJBs directly on the console. However you can use the console to display XML deployment descriptors for the enterprise application, Web modules and EJB modules.

To see the WAR files and JAR files associated with an enterprise application, do the following:

1. From the console navigation tree, select **Applications -> Enterprise Applications**.
2. Click the application that you are interested in.
3. Under the Configuration tab, select **View Deployment Descriptor** in the Additional Properties pane.

Figure 8-36 shows the deployment descriptor window for the DefaultApplication enterprise application. The Configuration tab shows you the structure defined by the deployment descriptor:

- ► The name and description of the enterprise application.
- ► The Web modules or WAR files and their context roots.
- ► The EJB modules and their associated JAR files.
- ► The security roles associated with the enterprise application.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Applicati
Install New Applicat
Resources
Security
Environment
System Administration
Troubleshooting
< Back
WebSphere Status
Local intranet

Application.xml - DefaultApplication.ear

Description: This is the IBM WebSphere Application Server Default Application.

- EJB Modules
  - EJB module name: **Increment.jar**
- Web Modules
  - Web Module URI: **DefaultWebApplication.war**
    - Context-root: /
- Security Roles
  - Role-name: **All Role**
    - Description: **All Authenticated users role.**

Figure 8-36 Enterprise application deployment descriptor

4. The Local Topology tab provides a view of the elements defined by the deployment descriptor. Each is a link to the configuration properties page for that element.

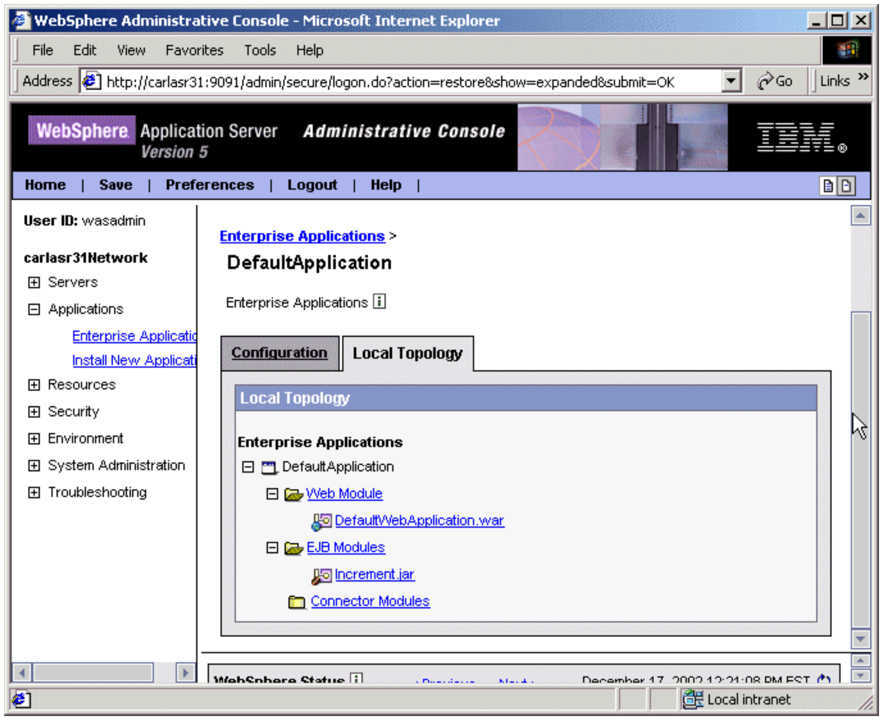
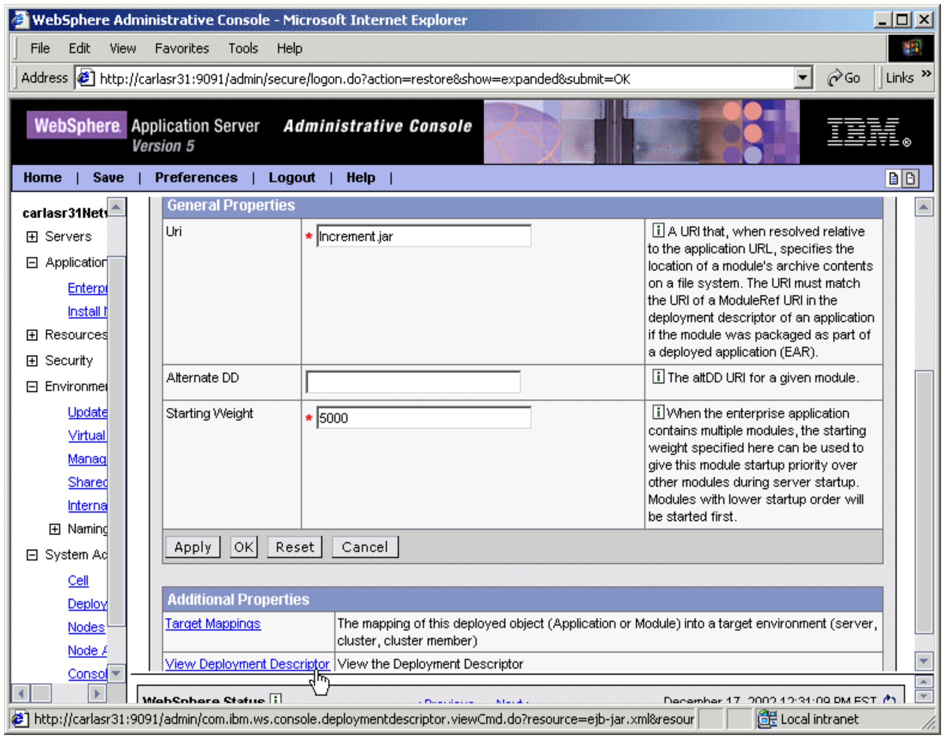


Figure 8-37 Topology view of the application

Viewing EJB modules

To see the EJBs that are part of an enterprise application:

1. Select **Applications -> Enterprise Applications**.
2. Click the application that you are interested in.
3. In the Related Items pane under the Configuration tab, select **EJB Modules**.
4. Click the EJB module you want to view.



WebSphere Administrative Console - Microsoft Internet Explorer

Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK

General Properties

Uri	* Increment.jar	A URI that, when resolved relative to the application URL, specifies the location of a module's archive contents on a file system. The URI must match the URI of a ModuleRef URI in the deployment descriptor of an application if the module was packaged as part of a deployed application (EAR).
Alternate DD		The altDD URI for a given module.
Starting Weight	* 5000	When the enterprise application contains multiple modules, the starting weight specified here can be used to give this module startup priority over other modules during server startup. Modules with lower startup order will be started first.

Additional Properties

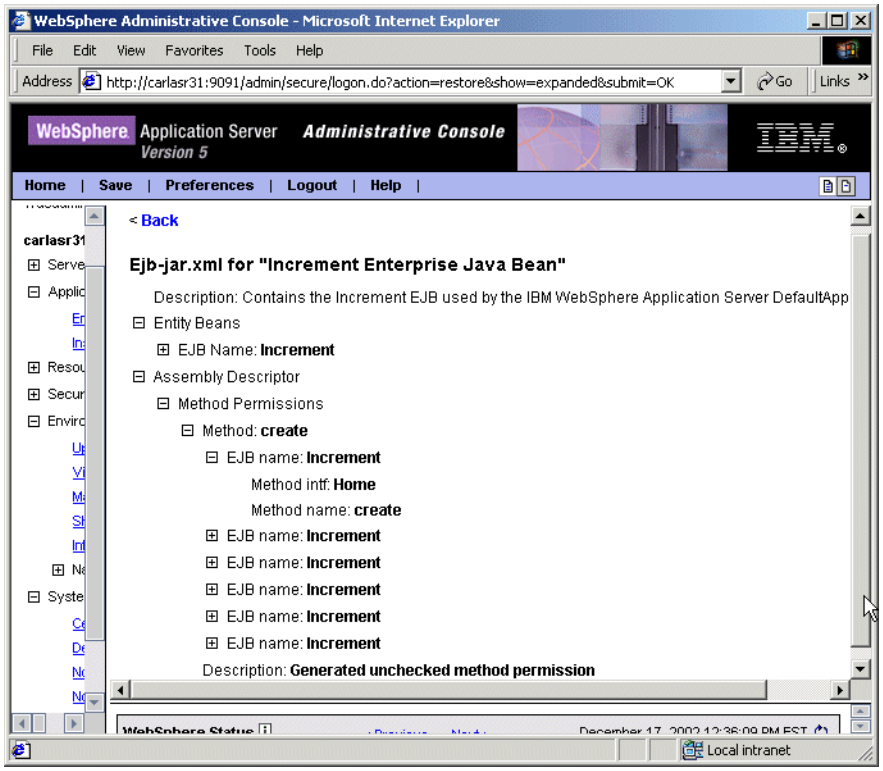
Target Mappings	The mapping of this deployed object (Application or Module) into a target environment (server, cluster, cluster member)
View Deployment Descriptor	View the Deployment Descriptor

http://carlasr31:9091/admin/com.ibm.ws.console.deploymentdescriptor.viewCmd.do?resource=ejb-jar.xml&resour

Local intranet

Figure 8-38 Viewing an EJB module configuration

5. Click **View Deployment Descriptor** in the Additional Properties pane to see the EJB deployment descriptor.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
carlasr31
Local intranet

< **Back**

Ejb-jar.xml for "Increment Enterprise Java Bean"

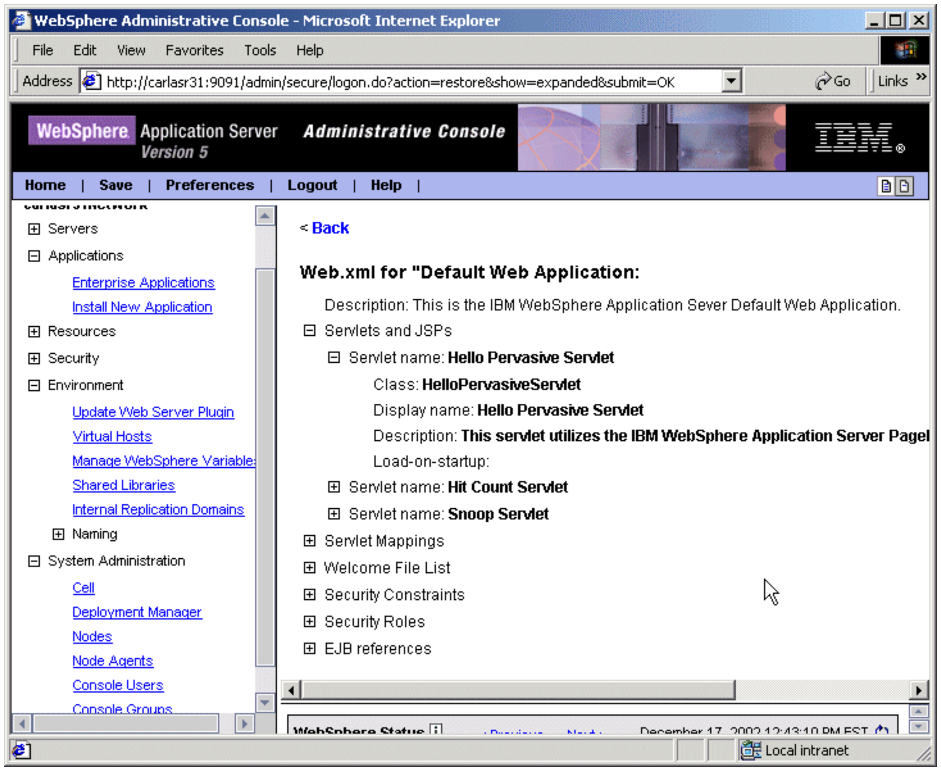
Description: Contains the Increment EJB used by the IBM WebSphere Application Server DefaultApp
- Entity Beans
  - EJB Name: **Increment**
- Assembly Descriptor
  - Method Permissions
    - Method: **create**
      - EJB name: **Increment**
        - Method intf: **Home**
        - Method name: **create**
      - EJB name: **Increment**
      - EJB name: **Increment**
      - EJB name: **Increment**
      - EJB name: **Increment**
      - EJB name: **Increment**
    - Description: **Generated unchecked method permission**

Figure 8-39 EJB module deployment descriptor

Viewing Web modules

To see the servlets and JSPs that are part of an enterprise application:

1. Select **Applications -> Enterprise Applications**.
2. Click the application that you are interested in.
3. Under the Related Items pane in the Configuration tab, select **Web Modules**.
4. Click the Web module you want to view.
5. Click **View Deployment Descriptor**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Servers
Applications
Enterprise Applications
Install New Application
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variables
Shared Libraries
Internal Replication Domains
Naming
System Administration
Cell
Deployment Manager
Nodes
Node Agents
Console Users
Console Groups
< Back
Local intranet

Web.xml for "Default Web Application:

Description: This is the IBM WebSphere Application Sever Default Web Application.
Servlets and JSPs
Servlet name: **Hello Pervasive Servlet**
Class: **HelloPervasiveServlet**
Display name: **Hello Pervasive Servlet**
Description: **This servlet utilizes the IBM WebSphere Application Server PageI**
Load-on-startup:
Servlet name: **Hit Count Servlet**
Servlet name: **Snoop Servlet**
Servlet Mappings
Welcome File List
Security Constraints
Security Roles
EJB references

Figure 8-40 Web module deployment descriptor

Finding a URL for a servlet or JSP

The URL for a servlet or JSP is the path used to access it from a browser. The URL is partly defined in the deployment descriptor provided in the EAR file and partly defined in the deployment descriptor for the Web module containing the servlet or JSP.

To find the URL for a servlet or JSP:

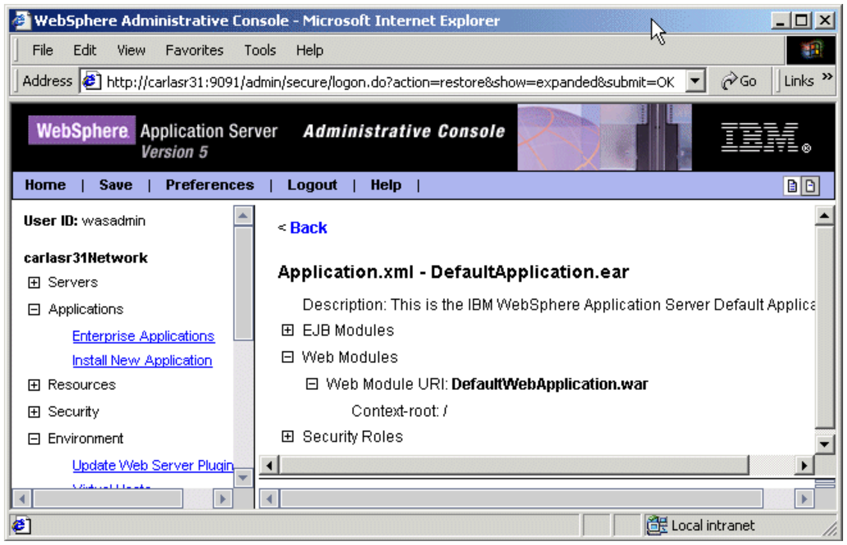
1. Find the context root of the Web module containing the servlet.
2. Find the URL for the servlet.
3. Find the virtual host where the Web module is installed.
4. Find the aliases by which the virtual host is known.
5. Combine the virtual host alias, context root, and URL pattern to form the URL request of the servlet/JSP.

For example, to look up the URL for the snoop servlet:

1. Find the context root of the Web module DefaultWebApplication of the DefaultApplication enterprise application. This Web module contains the snoop servlet.
   a. From the console navigation tree, select **Applications -> Enterprise Applications**.
   b. Click the application that you are interested in, in our case **DefaultApplication**.
   c. On the Configuration tab, select **View Deployment Descriptor** in the Additional Properties pane.

      Figure 8-41 shows the deployment descriptor window for the DefaultApplication enterprise application. You can see:

      i. There is only one Web module in this application, DefaultWebApplication.
      ii. The context root for the DefaultWebApplication Web module is "/".We will use this later.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Applications
Install New Application
Resources
Security
Environment
Update Web Server Plugin
< Back
Local intranet

Application.xml - DefaultApplication.ear

Description: This is the IBM WebSphere Application Server Default Applica
EJB Modules
Web Modules
Web Module URI: DefaultWebApplication.war
Context-root: /
Security Roles

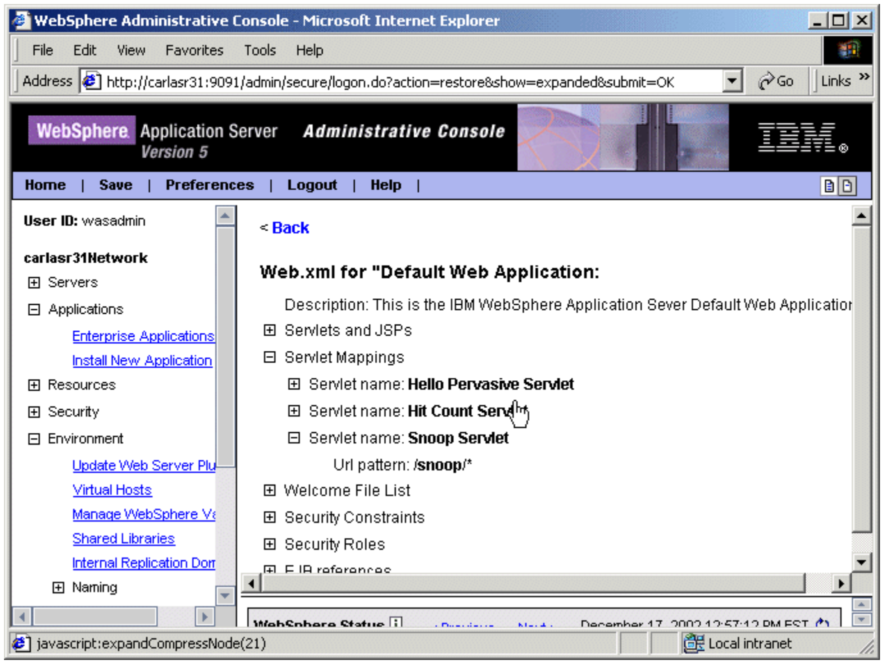
Figure 8-41 Deployment descriptor for DefaultApplication

   d. Click **Back** to return to the DefaultApplication configuration.
2. Find the URL for the snoop servlet:
   a. In the DefaultApplication configuration page, select **Web Modules** in the Related Items pane.

b. Click the DefaultWebApplication Web module to see the general properties.

c. Click **View Deployment Descriptor** in the Additional Properties pane.

This displays the Web module properties window, as shown in Figure 8-42. Note that the URL pattern for the snoop servlet starting from the Web module context root is “/snoop/*”. The Web module context root was “/”.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Applications
Install New Application
Resources
Security
Environment
Update Web Server Plu
Virtual Hosts
Manage WebSphere Va
Shared Libraries
Internal Replication Dom
Naming
< Back

Web.xml for "Default Web Application:

Description: This is the IBM WebSphere Application Sever Default Web Applicatio

⊞ Servlets and JSPs

⊟ Servlet Mappings

⊞ Servlet name: **Hello Pervasive Servlet**

⊞ Servlet name: **Hit Count Serv**

⊟ Servlet name: **Snoop Servlet**

Url pattern: **/snoop/***

⊞ Welcome File List

⊞ Security Constraints

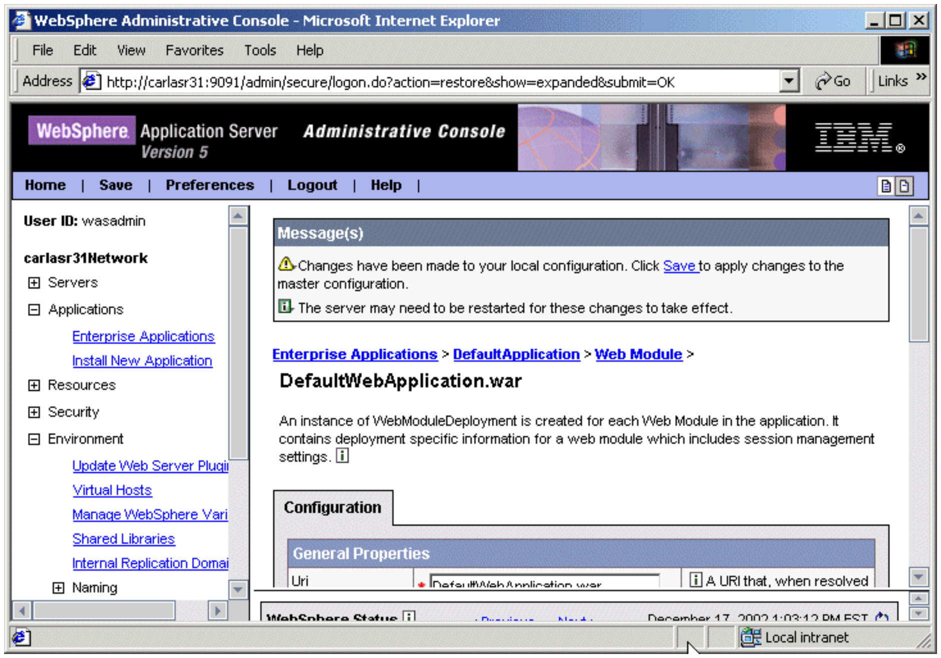
⊞ Security Roles

javascript:expandCompressNode(21)
Local intranet

Figure 8-42 DefaultWebApplication Web module deployment descriptor

d. Click **Back** to return to the Web module configuration page.

e. Note that as you navigate through the windows, a navigation path is displayed below the Messages area. Click **DefaultApplication** to return to the configuration page.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Applications
Install New Application
Resources
Security
Environment
Update Web Server Plugi
Virtual Hosts
Manage WebSphere Vari
Shared Libraries
Internal Replication Domai
Naming
Local intranet

Message(s)

Changes have been made to your local configuration. Click Save to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

Enterprise Applications > DefaultApplication > Web Module >

DefaultWebApplication.war

An instance of WebModuleDeployment is created for each Web Module in the application. It contains deployment specific information for a web module which includes session management settings.

Configuration

General Properties

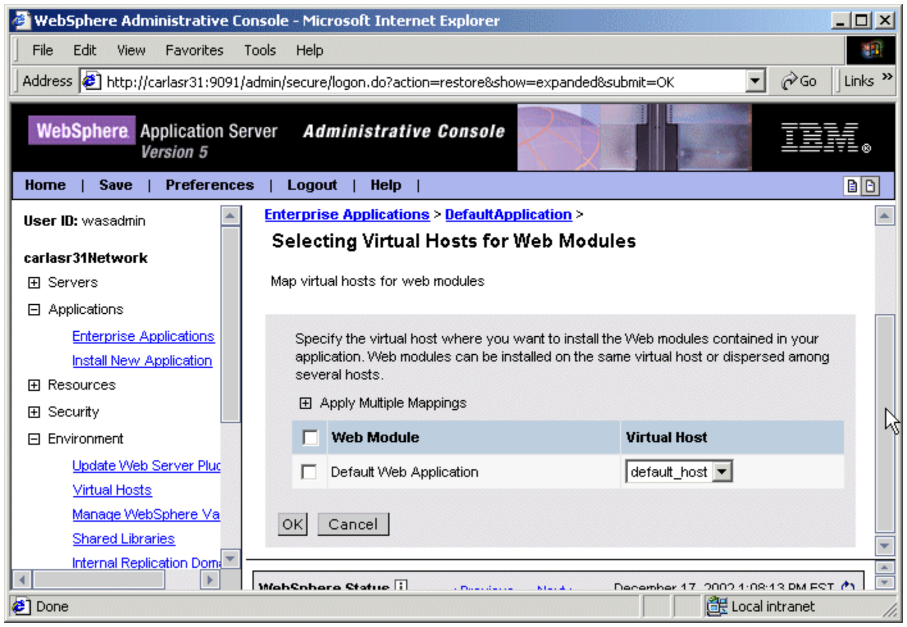
Uri	DefaultWebApplication.war	A URI that, when resolved

Figure 8-43 Return path

3. Find the virtual host where the DefaultWebApplication Web module is installed:

   a. In the DefaultApplication configuration page, select **Map virtual hosts for web modules** in the Additional Properties pane.

      This will display all of the Web modules contained in the enterprise application, and the virtual hosts in which they have been installed. Note that the DefaultWebApplication Web module has been installed on the default_host virtual host.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do?action=restore&show=expanded&submit=OK
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Enterprise Applications
Install New Application
Resources
Security
Environment
Update Web Server Pluc
Virtual Hosts
Manage WebSphere Va
Shared Libraries
Internal Replication Dom
Done
Local intranet

Enterprise Applications > DefaultApplication >

Selecting Virtual Hosts for Web Modules

Map virtual hosts for web modules

Specify the virtual host where you want to install the Web modules contained in your application. Web modules can be installed on the same virtual host or dispersed among several hosts.

Apply Multiple Mappings

	Web Module	Virtual Host
☐	Default Web Application	default_host

OK Cancel

Figure 8-44 List of virtual hosts

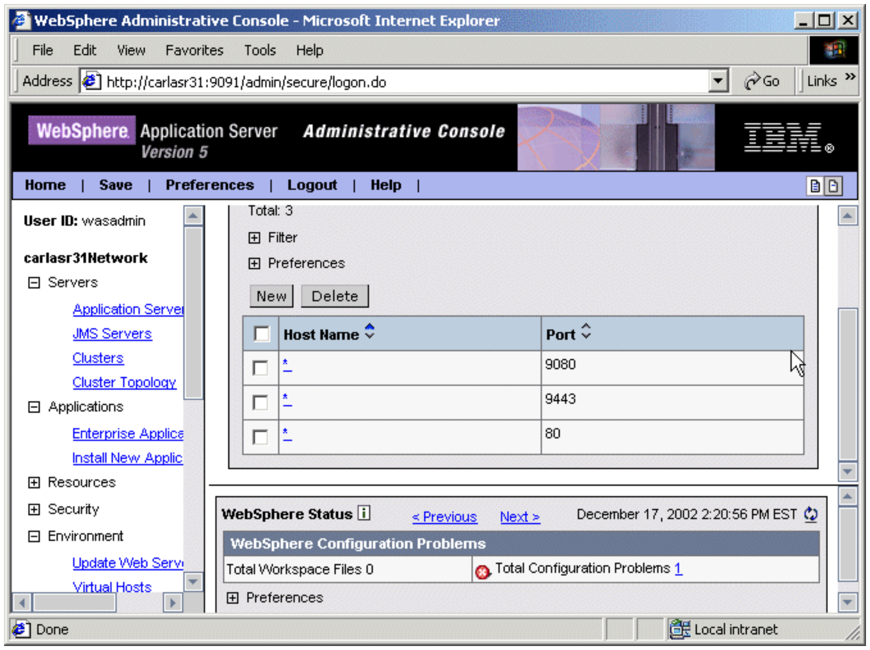
4. Find the host aliases for the default_host virtual host.

   a. From the console navigation tree, select **Environment -> Virtual Hosts**.

   b. Click **default_host**.

   c. Select **Host Aliases** in the Additional Properties pane.

      This shows the list of aliases by which the default_host virtual host is known.



	Host Name	Port
☐	*	9080
☐	*	9443
☐	*	80

Figure 8-45 Default_host virtual host aliases

Note that the aliases are composed of a DNS host name and a port number. The host aliases for the default_host virtual host are *:80, *:9080 and *:9443, “*” meaning any host name.

5. Combine the virtual host alias, context root and URL pattern to form the URL request of the snoop servlet. Requests for the servlet with any of the following URLs will map to the default_host virtual host:

```
http://<hostname>:80/snoop
http://<hostname>:9080/snoop
https://<hostname>:9443/snoop
```

8.6.7 Managing shared libraries

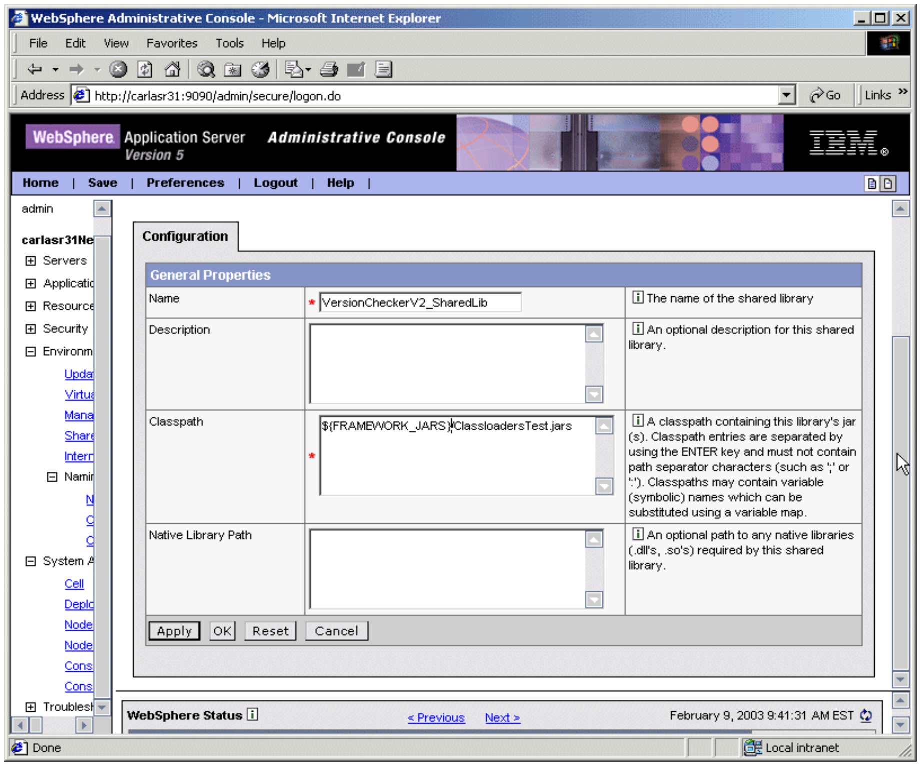
Shared libraries for use by deployed applications can be defined. By default, a shared library is accessible to applications deployed (or installed) on the same node as the shared library file. Use the Scope field to change the scope to a different node or to a specific server.

A shared library definition consists of a name, classpath entries to the library JAR files, and optionally, paths to any native libraries required by the shared library.

Understanding shared libraries: This section tells you how to configure shared libraries using the administrative console. For an example of using shared libraries, see, 14.7.4, “Step 4: Sharing dependency JARs among multiple applications” on page 702.

To define a shared library:

1. Select **Environment -> Shared Libraries**
2. Select the scope (cell, node, server) for the shared library definition.
3. Click **New**.
4. Specify the required properties. At minimum, this includes the name and classpath properties.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
admin
carlasr31Ne
Servers
Applicatio
Resource
Security
Environm
Upda
Virtu
Mana
Share
Intern
Namin
System A
Cell
Deplo
Node
Node
Cons
Cons
Troublesh
Configuration
WebSphere Status
< Previous Next >
February 9, 2003 9:41:31 AM EST
Done
Local intranet

General Properties

Name	* VersionCheckerV2_SharedLib	The name of the shared library
Description		An optional description for this shared library.
Classpath	* ${FRAMEWORK_JARS}/ClassloadersTest.jars	A classpath containing this library's jar(s). Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map.
Native Library Path		An optional path to any native libraries (.dll's, .so's) required by this shared library.

Apply OK Reset Cancel

Figure 8-46 Defining shared libraries

When defining the classpath, use the enter key to separate entries. Do not use any usual separators, such as “;” or “:”. The use of variables can be useful, but make sure to define the variables at the same scope as the shared library.

5. Click **OK**.

The next step is to add the library to the application.

1. Click **Applications -> Enterprise Applications**
2. Select the application, and in the additional properties pane select **Libraries**.
3. Click **Add**. You should see the library you just created (if not, you probably have a scoping problem).
4. Select the shared library, and click **OK**.
5. Save the configuration.

8.6.8 Generating the Web server plug-in

After certain changes in the WebSphere configuration, Web server plug-in should be updated so that Web server can handle requests properly. These changes include installing an application, creating or changing a virtual host, creating a new server, modifying HTTP transport settings, and creating or altering a cluster. As you go through this section, you will see an example of a plug-in file and the changes that would affect it should become apparent.

Generating the Web server plug-in

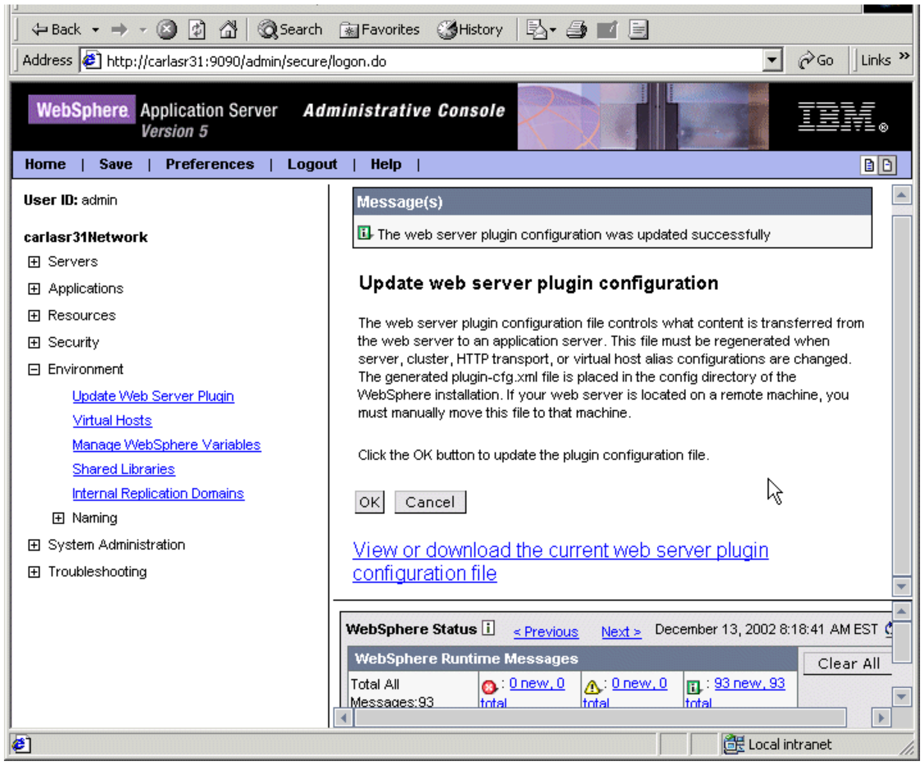
After the installation of the Web server and WebSphere Application Server, the Web server plug-in must be generated and placed in a location where the Web server can find it. Any future changes that would affect the way requests are routed would require that this process be repeated.

Understanding the Web server plug-in: This section tells you how to generate the Web server plug-in using the administrative console. For additional information about the Web server plug-in, see the following:

- For general information on the Web server plug-in, see “Web server plug-ins” on page 39.
- For information on regenerating the Web server plug-in with wsadmin, see 16.4.8, “Generating the Web server plug-in configuration” on page 809.
- For information on regenerating the Web server plug-in using the command line, see A.17, “GenPluginCfg” on page 874.

To generate the configuration file:

1. Open the administrative console.
2. Expand **Environment** and select **Update Web Server Plugin**.
3. Click **OK** to generate the configuration file.
4. Verify the generation was successful by looking at the messages.



http://carlasr31:9090/admin/secure/logon.do

Message(s)

The web server plugin configuration was updated successfully

Update web server plugin configuration

The web server plugin configuration file controls what content is transferred from the web server to an application server. This file must be regenerated when server, cluster, HTTP transport, or virtual host alias configurations are changed. The generated plugin-cfg.xml file is placed in the config directory of the WebSphere installation. If your web server is located on a remote machine, you must manually move this file to that machine.

Click the OK button to update the plugin configuration file.

View or download the current web server plugin configuration file

WebSphere Status < Previous Next > December 13, 2002 8:18:41 AM EST

WebSphere Runtime Messages

Total All Messages:93 : 0 new, 0 total : 0 new, 0 total : 93 new, 93 total

Figure 8-47 Generate the Web server plug-in

The generated Web server plug-in is stored in one of the following places:

– In a Network Deployment environment, it is stored in <WAS_ND_HOME>/config/cells/plugin-cfg.xml.

– In a base WebSphere Application Server environment, it is stored in <WAS_HOME>/config/cells/plugin-cfg.xml

You can view the file using the link on the administrative window or by opening it with a text editor. Example 8-1 on page 332 shows an excerpt from a generated file.

Example 8-1 Generated configuration file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config>
    <Log LogLevel="Error"
Name="C:\WebSphere\DeploymentManager\logs\http_plugin.log"/>
    <VirtualHostGroup Name="default_host">
        <VirtualHost Name="*:9080"/>
        <VirtualHost Name="*:9443"/>
        <VirtualHost Name="*:80"/>
    </VirtualHostGroup>

<ServerCluster Name="server1_carlasr31_Cluster">
        <Server Name="server1">
            <Transport Hostname="carlasr31" Port="9080" Protocol="http"/>
            <Transport Hostname="carlasr31" Port="9443" Protocol="https">
                <Property name="keyring"
value="C:\WebSphere\DeploymentManager\etc\plugin-key.kdb"/>
                <Property name="stashfile"
value="C:\WebSphere\DeploymentManager\etc\plugin-key.sth"/>
            </Transport>
        </Server>
        <PrimaryServers>
            <Server Name="server1"/>
        </PrimaryServers>
    </ServerCluster>

   <UriGroup Name="default_host_server1_carlasr31_Cluster_URIs">
        <Uri AffinityCookie="JSESSIONID" Name="/webbank/*"/>
    </UriGroup>
    <Route ServerCluster="server1_carlasr31_Cluster"
        UriGroup="default_host_server1_carlasr31_Cluster_URIs"
VirtualHostGroup="default_host"/>
</Config>
```

The specific values for the UriGroup Name and AffinityCookie attributes will depend on how you have assembled your application. When you assemble your application, if you specify “File Serving Enabled” then only a wildcard URI is generated, regardless of any explicit servlet mappings. If you specify “serve servlets by classmate” then a URI of the form “URI name = "<webapp uri>/servlet/*” is generated. This applies for both the Name and AffinityCookie attributes.

5. Open the file and alter directory paths if necessary. The plugin-cfg.xml contains paths to files. These paths were generated with respect to the

WebSphere Application Server system where the plug-in was generated and may not be correct for the target Web server plug-in installation.

The configuration file in Example 8-1 on page 332 has the Deployment Manager installation directory as a base for the log file and for the keyring and stashfile values. If you assume that the Web server plug-in was installed under the WebSphere Application Server directory on the Web server you will need to change the configuration file to reflect that.

Example 8-2 Generated configuration file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Config>
    <Log LogLevel="Error" Name="C:\WebSphere\AppServer\logs\http_plugin.log"/>
    <VirtualHostGroup Name="default_host">
        <VirtualHost Name="*:9080"/>
        <VirtualHost Name="*:9443"/>
        <VirtualHost Name="*:80"/>
    </VirtualHostGroup>

<ServerCluster Name="server1_carlasr31_Cluster">
        <Server Name="server1">
            <Transport Hostname="carlasr31" Port="9080" Protocol="http"/>
            <Transport Hostname="carlasr31" Port="9443" Protocol="https">
                <Property name="keyring"
value="C:\WebSphere\AppServer\etc\plugin-key.kdb"/>
                <Property name="stashfile"
value="C:\WebSphere\AppServer\etc\plugin-key.sth"/>
            </Transport>
        </Server>
        <PrimaryServers>
            <Server Name="server1"/>
        </PrimaryServers>
    </ServerCluster>

   <UriGroup Name="default_host_server1_carlasr31_Cluster_URIs">
        <Uri AffinityCookie="JSESSIONID" Name="/webbank/*"/>
    </UriGroup>
    <Route ServerCluster="server1_carlasr31_Cluster"
        UriGroup="default_host_server1_carlasr31_Cluster_URIs"
VirtualHostGroup="default_host"/>
</Config>
```

6. The configuration file must reside in the location specified in the Web server configuration file.

   For the IBM HTTP Server, the configuration file is stored in <<IHS_HOME>>/config/httpd.conf. Open the file and find the line for the Web server plug-in:

   ```
   WebSpherePluginConfig "C:\WebSphere\AppServer/config/cells/plugin-cfg.xml"
   ```

7. The location is determined by the directory you specified for WebSphere during the Web server plug-in install.

   Verify that the location is correct and copy the file to this location on the Web server machine. You will need to restart the Web server for the changes to take place.

Note: If you have just installed the Web server plug-in, check the Web server configuration file to make sure the appropriate statements have been added to point to the plug-in. In particular, make sure the WebSpherePluginConfig directory is correct for your installation. This is where you will copy the plug-in file every time you generate it.

For the IBM HTTP Server, you will see the following in <IHS_HOME>/conf/httpd.conf:

```
Alias /WSsamples /usr/WebSphere/AppServer/WSsamples
Alias /IBMWebAS/ /usr/WebSphere/AppServer/web/
LoadModule ibm_app_server_http_module
/usr/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /usr/WebSphere/AppServer/config/plugin-cfg.xml
```

Restarting the Web server

The Web server plug-in will check for a new configuration file every 60 seconds. You can wait for the plug-in to find the changes, or you can restart the Web server to pick up the changes earlier.

9

Session management

Session support allows a Web application developer to maintain state information across multiple user visits to the application. This chapter discusses HTTP session support in WebSphere Application Server V5.1 and how to configure it.

We would recommend that you also read *IBM WebSphere V5.0 Performance, Scalability and High Availability*, SG24-6198.

9.1 Session management

In many Web applications, users dynamically collect data as they move through the site based on a series of selections on pages they visit. Where the user goes next, and what the application displays as the user's next page (or next choice) may depend on what the user has chosen previously from the site. For example, if the user clicks the checkout button on a site, the next page must contain the user's shopping selections.

In order to do this, a Web application needs a mechanism to hold the user's state information over a period of time. However, HTTP alone doesn't recognize or maintain a user's state. HTTP treats each user request as a discrete, independent interaction.

The Java servlet specification provides a mechanism for servlet applications to maintain a user’s state information. This mechanism, known as a session, addresses some of the problems of more traditional strategies such as a pure cookie solution. It allows a Web application developer to maintain all user state information at the host, while passing minimal information back to the user via cookies, or another technique known as URL rewriting.

9.2 Version 4 versus Version 5 session management

The following improvements have been made to session management of WebSphere V5:

- Session manager configuration enhancement: In WebSphere V4, session manager configuration is at the application server level. In WebSphere V5, it can be defined at the following levels:
  - Application server level
  - Application level
  - Web module level
- Session scope enhancement

  As per the Servlet 2.3 specification, session scope is defined per Web application. In WebSphere V5, the IBM extensions provide the option to have session scope defined per enterprise application.
- Session enhancements
  - Session affinity has been improved.
  - Access to the session can be serialized.

- Session persistence

  There is a new feature, WebSphere internal messaging, which is in-memory replication of HTTP session state between clustered Web application servers.

  Note: This new feature is only available for WebSphere Network Deployment environments.

- Session counters in Tivoli Performance Viewer: New session counters are added in the runtime to analyze the session manager runtime.

9.3 Session manager configuration

In WebSphere Application Server V4, session management configuration was at the application server level, meaning all applications used the same session manager settings. When an application would benefit from customized settings, such as session timeout or persistence, or from using its own database for persistence to reduce the load generated by multiple applications, the only way to achieve this was by putting it on a separate application server.

In WebSphere Application Server V5, this is no longer a problem. Session management can be defined at the following levels:

- Application server: This is the default level. Configuration at this level is applied to all Web modules within the server.
- Application: Configuration at this level is applied to all Web modules within the application.
- Web module: Configuration at this level is applied only to that Web module.

9.3.1 Session management properties

With one exception, the session management properties you can set are the same at each configuration level:

- Session tracking mechanism: Select from cookies, URL rewriting, and SSL ID tracking. Selecting cookies will lead you to a second configuration page containing further configuration options.
- Maximum in-memory session count and whether to allow this number to be exceeded (overflow).
- The amount of time to allow a session to remain idle before timing out.
- Security integration: Specifies that the user ID be associated with the HTTP session.

- Serialize session access: Determines if concurrent session access in a given server is allowed.
- Overwrite session management (enterprise application and Web module level only: Determines whether these session management settings are used for the current module, or if the settings are used from the parent object.
- Distributed environment settings: Select how to persist sessions (memory-to-memory replication or a database) and set tuning properties.

9.3.2 Accessing session management properties

All configuration settings can be done using the administrative console.

Application server session management properties

To access session management properties at the application server level from the administrative console:

1. Click **Servers -> Application Servers**.
2. Click the application server.
3. In the Additional Properties table of the Configuration tab, click **Web Container**.
4. In the Additional Properties table, click **Session Management**.

Application session management properties

To access session management properties at the application level from the administrative console:

1. Click **Applications -> Enterprise Applications**.
2. Click the application.
3. In the Additional Properties table of the Configuration tab, click **Session Management**.

Web module session management properties

To access session management properties at the Web module level from administrative console:

1. Click **Applications -> Enterprise Applications**.
2. Click the application.
3. In the Related Items table of the Configuration tab, click **Web Modules**.
4. Click the Web module.
5. In the Additional Properties table, click **Session Management**.

9.4 Session scope

The Servlet 2.3 specification defines session scope at the Web application level, meaning, session information can only be accessed by a single Web application. However, there may be times when there is a logical reason for multiple Web applications to share information, for example, a user name.

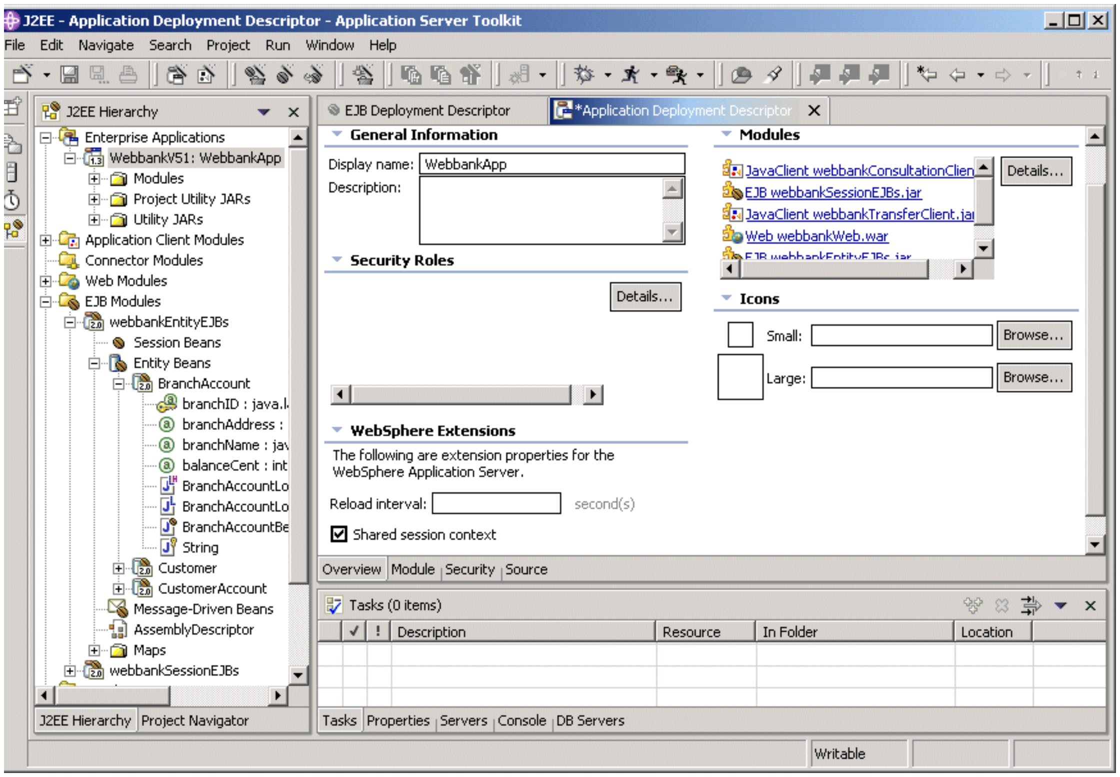
WebSphere V5 provides an IBM extension to the specification allowing session information to be shared among Web applications within an enterprise application. This option is offered as an extension to the application deployment descriptor. No code change is necessary to enable this option. This option is specified during application assembling.

Note: Since session information is shared within the enterprise application, you cannot use the Overwrite Session Management property at the Web module level when this option is selected.

Sharing session context

The WebSphere extension for sharing session context is set in the META-INF/ibm-application-ext.xmi file in the enterprise project. You can set this using the Assembly Toolkit or from WebSphere Studio:

1. Start the Assembly Toolkit or WebSphere Studio and switch to the J2EE perspective.
2. Double-click the EAR file in the J2EE Hierarchy view. This will open the application deployment descriptor.
3. Click the **Overview** tab.
4. Select **Shared session context**.



Display name: WebbankApp
Description:

The following are extension properties for the WebSphere Application Server.

Reload interval: second(s)

☑ Shared session context

Figure 9-1 Setting shared HTTP session context using the Assembly Toolkit

5. Save and close the deployment descriptor.

9.5 Session identifiers

WebSphere session support keeps the user's session information on the server. WebSphere passes the user an identifier known as a session ID, which correlates an incoming user request with a session object maintained on the server.

Note: The example session IDs provided in this chapter are for illustrative purposes only and are *not* guaranteed to be absolutely consistent in value, format and length.

9.5.1 Choosing a session tracking mechanism

WebSphere supports three approaches to track sessions:

- SSL session identifiers
- Cookies
- URL rewriting

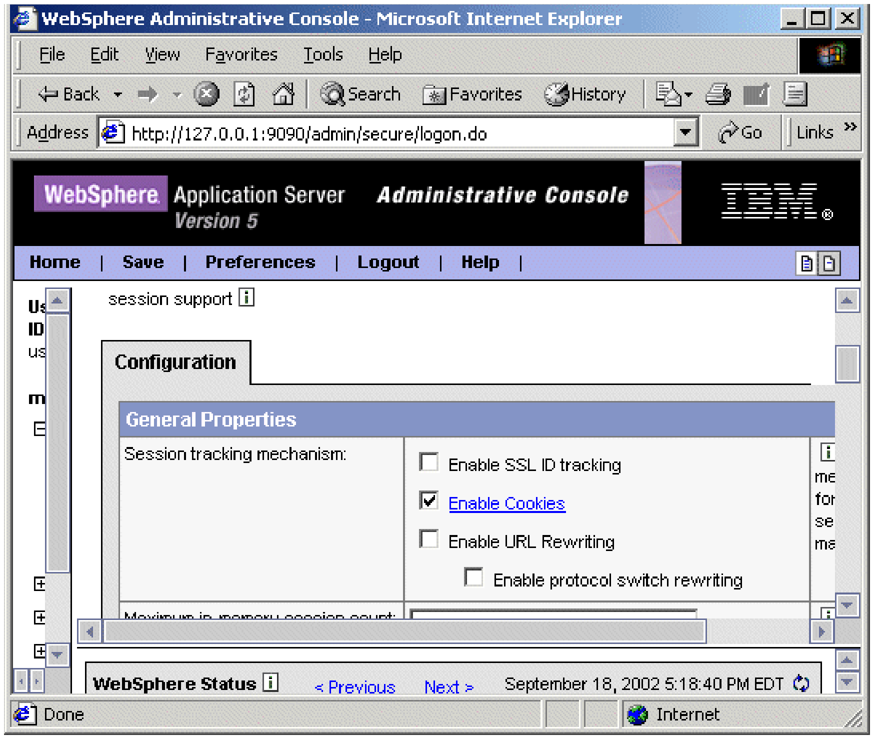
It is possible to select all three options for a Web application. If you do this:

- SSL session identifiers are used in preference to cookie and URL rewriting.
- Cookies are used in preference to URL rewriting.

Note: If SSL session ID tracking is selected, it is recommended that you also select cookies or URL rewriting so that session affinity can be maintained.

To set or change the session mechanism type:

1. Open the session management properties for the application server, enterprise application, or Web module.
2. Select the session tracking mechanism that you require.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
session support
Configuration
General Properties

Session tracking mechanism:
- [ ] Enable SSL ID tracking
- [✓] Enable Cookies
- [ ] Enable URL Rewriting
  - [ ] Enable protocol switch rewriting

WebSphere Status
< Previous
Next >
September 18, 2002 5:18:40 PM EDT
Done
Internet

Figure 9-2 Selecting a session tracking mechanism window

3. Click **OK**.
4. Save and synchronize the configuration changes.
5. Restart the application server or the cluster.

9.5.2 SSL ID tracking

When SSL ID tracking is enabled for requests over SSL. SSL session information is used to track the HTTP session ID.

Because the SSL session ID is negotiated between the Web browser and HTTP server, it cannot survive an HTTP server failure. However, the failure of an application server does not affect the SSL session ID. Of course, if session persistence is not configured, the session itself is lost. In environments that use WebSphere Edge Server with multiple HTTP servers, an affinity mechanism must be used when SSL session ID is to be used as the session tracking mechanism.

SSL tracking is supported only for the IBM HTTP Server and SUN ONE Web Server.

The lifetime of an SSL session ID can be controlled by configuration options in the Web server. For example, in the IBM HTTP Server, the configuration variable SSLV3TIMEOUT must be set to allow for an adequate lifetime for the SSL session ID. Too short an interval could result in premature termination of a session. Also, some Web browsers might have their own timers that affect the lifetime of the SSL session ID. These Web browsers might not leave the SSL session ID active long enough to be useful as a mechanism for session tracking.

When the SSL session ID is to be used as the session tracking mechanism in a clustered environment, either cookies or URL rewriting must be used to maintain session affinity. The cookie or rewritten URL contains session affinity information that enables the Web server to properly route requests back to the same server once the HTTP session has been created on a server. The SSL ID is not sent in the cookie or rewritten URL but is derived from the SSL information.

Disadvantages of SSL ID tracking

The main disadvantage of using SSL ID tracking is the performance hit of using SSL. If you have a business requirement to use SSL, then this would be a good choice. If you don't have such a requirement, it is probably a good idea to consider using cookies instead.

As discussed previously, Web server and Web browser SSL session timeout settings may also limit the usefulness of SSL ID tracking.

9.5.3 Cookies

Many sites choose cookie support to pass the user's identifier between WebSphere and the user. WebSphere Application Server session support generates a unique session ID for each user, and returns this ID to the user's browser via a cookie. The default name for the session management cookie is JSESSIONID.

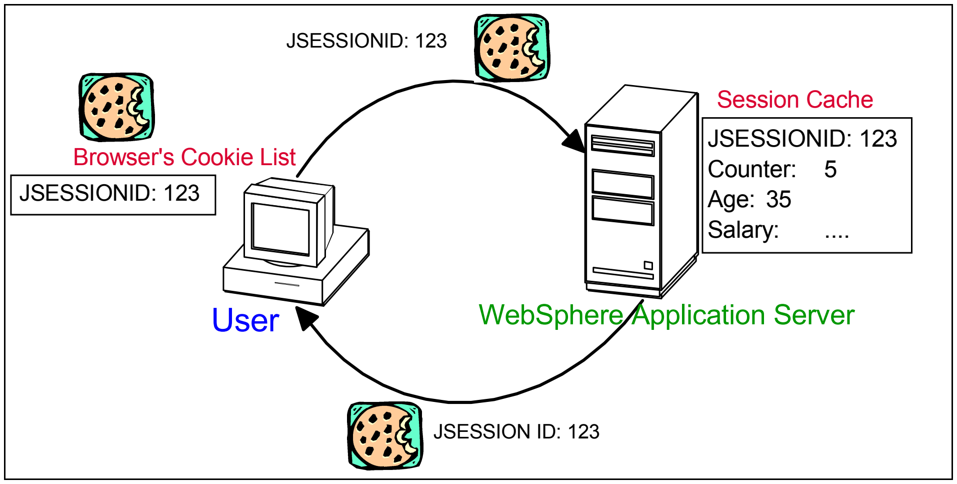


Figure 9-3 Cookie overview

A cookie consists of information embedded as part of the headers in the HTML stream passed between the server and the browser. The browser holds the cookie and returns it to the server whenever the user makes a subsequent request. By default, WebSphere defines its cookies so they are destroyed if the browser is closed. This cookie holds a session identifier. The remainder of the user's session information resides at the server.

The Web application developer uses the HTTP request object's standard interface to obtain the session:

```
HttpSession session = request.getSession(true);
```

WebSphere places the user's session identifier in the outbound cookie whenever the servlet completes its execution, and the HTML response stream returns to the end user. Again, neither the cookie nor the session ID within it require any direct manipulation by the Web application. The Web application only sees the contents of the session.

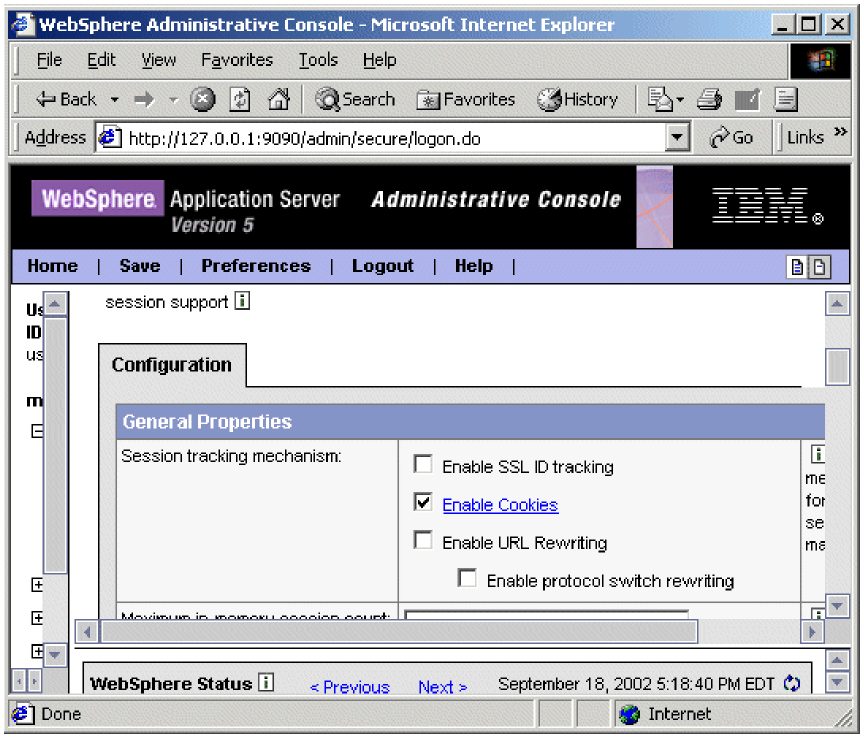
Cookie disadvantages

The main disadvantage with cookies is that some users, either by choice or mandate, disable them from within their browser.

Cookie settings

To configure session management using cookies from administrative console:

1. Open the Session Manager window at your preferred level.
2. Select **Enable Cookies** as the session tracking mechanism (check the box).


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
WebSphere Status
< Previous Next >
September 18, 2002 5:18:40 PM EDT
Done
Internet

session support

Configuration

General Properties

Session tracking mechanism:

- ☐ Enable SSL ID tracking
- ☑ Enable Cookies
- ☐ Enable URL Rewriting
  - ☐ Enable protocol switch rewriting

Figure 9-4 Session tracking mechanism

3. Select the **Enable Cookies** box. If you would like to view or change the cookies settings, select the **Enable Cookies** hot link. The following cookie settings are available:

   – Cookie name

     The cookie name for session management should be unique. The default cookie name is JSESSIONID, which is required by the Servlet 2.3 specification for all cookie-based session IDs. However, this value can be configured for flexibility.

   – Secure cookies

     Enabling the feature will restrict the exchange of cookies only to HTTPS sessions. If it is enabled the session cookie's body includes the "secure" indicator field.

   – Cookie domain

     This value will dictate to the browser whether or not to send a cookie to particular servers. For example, if you specify a particular domain, the browser will only send back session cookies to hosts in that domain. The default value in the session manager restricts cookies to the host that sent them.

Note: The LTPA token/cookie that is sent back to the browser is scoped by a single DNS domain that is specified when global security is configured. This means that *all* application servers in an *entire* WebSphere Application Server domain must share the same DNS domain for security purposes.

– Cookie path

  The paths (on the server) to which the browser will send the session tracking cookie. Specify any string representing a path on the server. Use "/" to indicate the root directory.

  Specifying a value will restrict the paths to which the cookie will be sent. By restricting paths, you can keep the cookie from being sent to certain URLs on the server. If you specify the root directory, the cookie will be sent no matter which path on the given server is accessed.

– Cookie maximum age

  The amount of time that the cookie will live in the client browser. There are two choices:

  - Expire at the end of the current browser session
  - Expire at a configurable maximum age

  If you choose the maximum age option, specify the age in seconds.

4. Click **OK** to exit the page and change your settings.
5. Click **OK** to exit the session management settings.
6. Save and synchronize your configuration changes.
7. Restart the application server or the cluster.

For more information on cookie properties, please refer to the following Web site:

http://home.netscape.com/newsref/std/cookie_spec.html

9.5.4 URL rewriting

WebSphere also supports URL rewriting for session ID tracking. While session management using SSL IDs or cookies is transparent to the Web application, URL rewriting requires the developer to use special encoding APIs, and to set up the site page flow to avoid losing the encoded information.

URL rewriting works by actually storing the session identifier in the page returned to the user. WebSphere encodes the session identifier as a parameter on URLs that have been encoded programmatically by the Web application developer. Example 9-1 shows a Web page link with URL encoding.

Example 9-1 Web page link with URL rewriting

```
<a href="/store/catalog;$jsessionid=DA32242SSGE2">
```

When the user clicks this link (Example 9-1) to move to the /store/catalog page, the session identifier passes into the request as a parameter.

URL rewriting requires explicit action by the Web application developer. If the servlet returns HTML directly to the requester (without using a JavaServer Page), the servlet calls the API, as shown in Example 9-2, to encode the returning content.

Example 9-2 URL encoding from a servlet

```
out.println("<a href=\");
out.println(response.encodeURL ("/store/catalog"));
out.println("\>catalog</a>");
```

Even pages using redirection (a common practice, particularly with servlet or JSP combinations) must encode the session ID as part of the redirect, as shown in Example 9-3.

Example 9-3 URL encoding with redirection

```
response.sendRedirect(response.encodeRedirectURL("http://myhost/store/catalog")
);
```

When JavaServer Pages (JSPs) use URL rewriting, the JSP calls a similar interface to encode the session ID, as shown in Example 9-4.

Example 9-4 URL encoding in a JSP

```
<% response.encodeURL ("/store/catalog"); %>
```

URL rewriting configuration

URL rewriting is selected in the same way as cookies. The only additional configuration option is:

- Enable protocol switch rewriting

  Defines whether the session ID, added to a URL as part of URL encoding, should be included in the new URL if a switch from HTTP to HTTPS or from HTTPS to HTTP is required. When URL encoding is enabled, this setting determines encoding of HTTPS URLs in HTTP requests, or encoding of

HTTP URLs in HTTPS requests. That is, if a servlet is accessed over HTTP and if that servlet is doing encoding of HTTPS URLs, URL encoding will be performed only when protocol switch rewriting is enabled, and vice versa.

Disadvantages of using URL rewriting

The fact that the servlet or JSP developer has to write extra code is a major drawback over the other available session tracking mechanisms.

URL rewriting limits the flow of site pages exclusively to dynamically generated pages (such as pages generated by servlets or JSPs). WebSphere inserts the session ID into dynamic pages, but cannot insert the user's session ID into static pages (.htm or .html pages).

Therefore, after the application creates the user's session data, the user must visit dynamically generated pages exclusively until they finish with the portion of the site requiring sessions. URL rewriting forces the site designer to plan the user's flow in the site to avoid losing their session ID.

9.6 Local sessions

Many Web applications use the simplest form of session management: the in-memory, local session cache. The local session cache keeps session information in memory and local to the machine and WebSphere Application Server where the session information was first created.

Local session management doesn't share user session information with other clustered machines. Users only obtain their session information if they return to the machine and WebSphere Application Server holds their session information on subsequent accesses to the Web site.

Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server, but also destroys any sessions managed by those instances.

WebSphere allows the administrator to define a limit on the number of sessions held in the in-memory cache via the administrative console settings on the session manager. This prevents the sessions from acquiring too much memory in the Java virtual machine associated with the application server.

The session manager also allows the administrator to permit an unlimited number of sessions in memory. If the administrator enables the **Allow overflow** setting on the session manager, the session manager permits two in-memory caches for session objects. The first cache contains only enough entries to accommodate the session limit defined to the session manager (1000 by default). The second cache, known as the overflow cache, holds any sessions the first cache cannot accommodate, and is limited in size only by available memory. The session manager builds the first cache for optimized retrieval, while a regular, un-optimized hash table contains the overflow cache.

For best performance, define a primary cache of sufficient size to hold the normal working set of sessions for a given Web application server.

Important: You should note that with overflow enabled, the session manager permits an unlimited number of sessions in memory. Without limits, the session caches may consume all available memory in the WebSphere instance’s heap, leaving no room to execute Web applications. For example, two scenarios under which this could occur are:

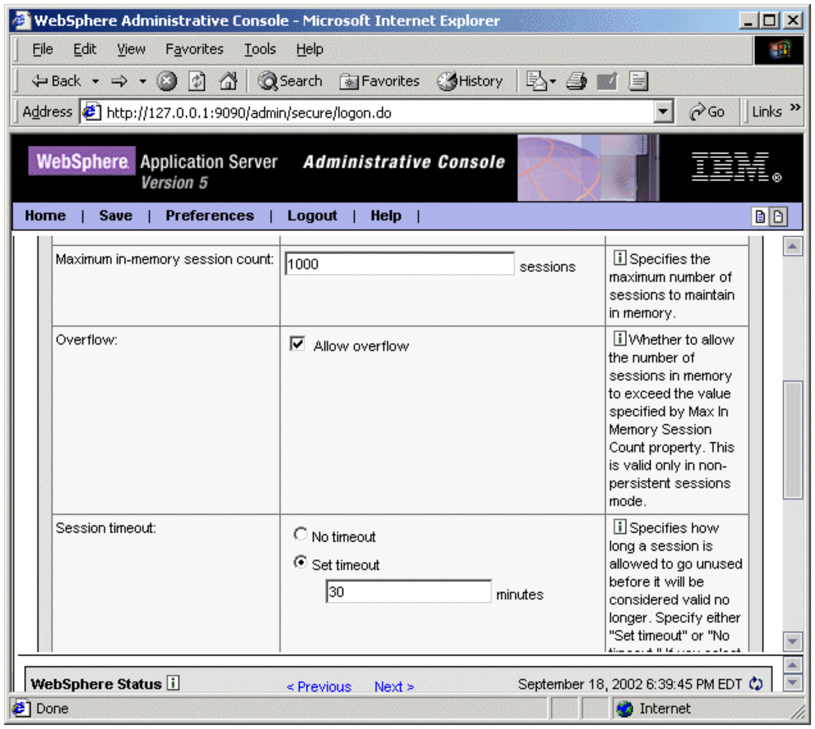
- The site receives greater traffic than anticipated, generating a large number of sessions held in memory.
- A malicious attack occurs against the site where a user deliberately manipulates their browser so the application creates a new session repeatedly for the same user.

If you choose to enable session overflow, the state of the session cache should be monitored closely.

Note: Each Web application will have its own base (or primary) in-memory session cache, and with overflow allowed, its own overflow (or secondary) in-memory session cache.

9.7 Advanced settings for session management

The session management settings allow the administrator to tune a number of parameters that are important for both local or persistent sessions:


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |

Maximum in-memory session count:	1000 sessions	Specifies the maximum number of sessions to maintain in memory.
Overflow:	☑ Allow overflow	Whether to allow the number of sessions in memory to exceed the value specified by Max In Memory Session Count property. This is valid only in non-persistent sessions mode.
Session timeout:	○ No timeout ● Set timeout 30 minutes	Specifies how long a session is allowed to go unused before it will be considered valid no longer. Specify either "Set timeout" or "No

Figure 9-5 Session Management window (2)

- Maximum in-memory session count

  Specifies the maximum number of sessions to maintain in memory.

  The meaning differs depending on whether you are using local or persistent sessions. For local sessions, this value specifies the number of sessions in the base session table. Select **Overflow** to specify whether to limit sessions to this number for the entire session manager, or to allow additional sessions to be stored in secondary tables. Before setting this value, see 9.6, “Local sessions” on page 348.

  For persistent sessions, this value specifies the size of the general cache. This value specifies how many session will be cached before the session manager reverts to reading a session from the database automatically. Session manager uses an LRU (least recently used) algorithm to maintain the sessions in the cache.

  This value holds when you are using local sessions, persistent sessions with caching, or persistent sessions with manual updates. The manual update cache keeps the last n time stamps representing “last access” times, where n is the maximum in-memory session count value.

- Overflow

  This specifies whether to allow the number of sessions in memory to exceed the value specified in the Maximum in-memory session count field. If Allow overflow is not checked, then WebSphere will limit the number of sessions held in memory to this value.

  For local sessions, if this maximum is exceeded and Allow overflow is not checked, then sessions created thereafter will be dummy sessions and will not be stored in the session manager. Before setting this value, see 9.6, “Local sessions” on page 348.

  As shown in Example 9-5, the IBM HttpSession extension can be used to react if sessions exceed the maximum number of sessions specified when overflow is disabled.

Example 9-5 Using IBMSession to react to session overflow

```
com.ibm.websphere.servlet.session.IBMSession sess =
   (com.ibm.websphere.servlet.session.IBMSession) req.getSession(true);
if(sess.isOverFlow()) {
   //Direct to a error page…
}
```

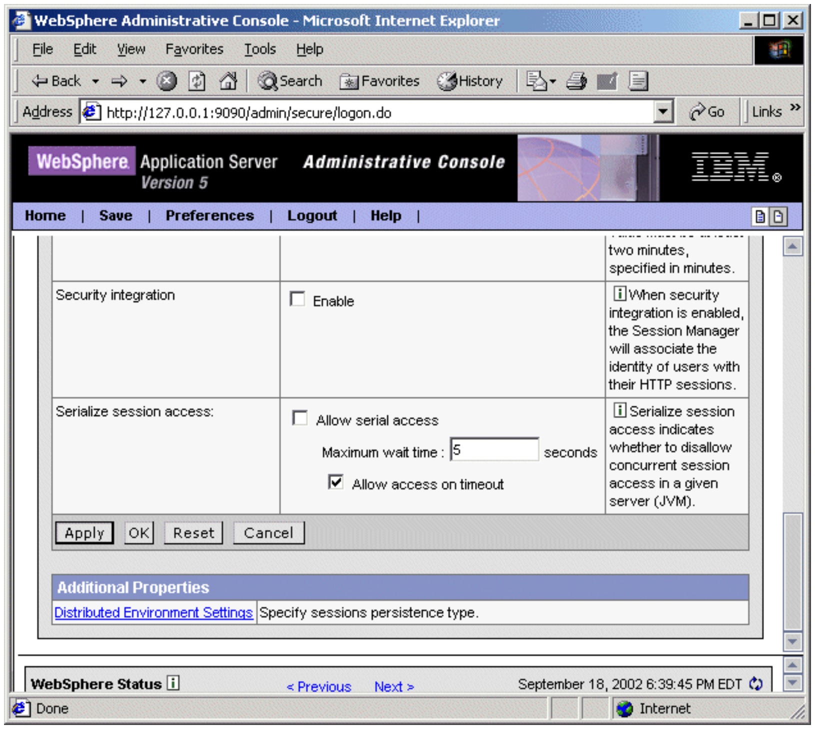
- Session timeout

  If **set timeout** is selected, when a session isn’t accessed for this many minutes it can be removed from the in-memory cache and if persistent sessions are used, from the persistent store. This is important for performance tuning. It directly influences the amount of memory consumed by the JVM in order to cache the session information.

  > **Note:** For performance reasons, the invalidation timer is not accurate “to the second.” It is safe to assume that the timer is accurate to within two minutes. When the write frequency is time-based, this value should be at least twice as large as the write interval.

  The value of this setting is used as a default when the session timeout is not specified in a Web module’s deployment descriptor.

  If **no timeout** is selected, a session will be never removed from the memory unless explicit invalidation has been performed by the servlet. This may cause memory leak when the user closes the window without performing logout from the system. This option might be useful when sessions should be kept for a while until explicit invalidation has been done, for example, an employee leaves the company. In order to use this option, it is necessary to make sure that enough memory or space in a persistent store is kept to accommodate all sessions.



		two minutes, specified in minutes.
Security integration	☐ Enable	When security integration is enabled, the Session Manager will associate the identity of users with their HTTP sessions.
Serialize session access:	☐ Allow serial access Maximum wait time : 5 seconds ☑ Allow access on timeout	Serialize session access indicates whether to disallow concurrent session access in a given server (JVM).

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://127.0.0.1:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help
Apply OK Reset Cancel
Additional Properties
WebSphere Status
< Previous Next >
September 18, 2002 6:39:45 PM EDT
Done
Internet

Additional Properties	
Distributed Environment Settings	Specify sessions persistence type.

Figure 9-6 Session Management window (3)

- Security integration

  When security integration is enabled, the session manager will associate the identity of users with their HTTP sessions. See 9.11, "Session security" on page 389 for more information.

> **Note:** Do not enable this property if the application server contains a Web application that has form-based login configured as the authentication method and the local operating system is the authentication mechanism. It will cause authorization failures when user agents try to use the Web application.

- Serialize session access

  In WebSphere V4, sessions can be accessed concurrently, meaning multiple threads can access the same session at the same time. It is the programmer's responsibility to serialize access to the session to avoid inconsistencies.

In WebSphere V5, there is an option to provide serialized access to the session in a given JVM. This ensures thread-safe access when the session is accessed by multiple threads. No special code is necessary for using this option. This option is not recommended when framesets are used heavily because it can affect performance.

An optional property, **Maximum wait time,** can be set to specify the maximum amount of time a servlet request waits on an HTTP session before continuing execution. The default is 2 minutes.

If the **Allow access on timeout** option is set, multiple servlet requests that have timed out concurrently will execute concurrently. If it is false, servlet execution aborts.

9.8 Session affinity

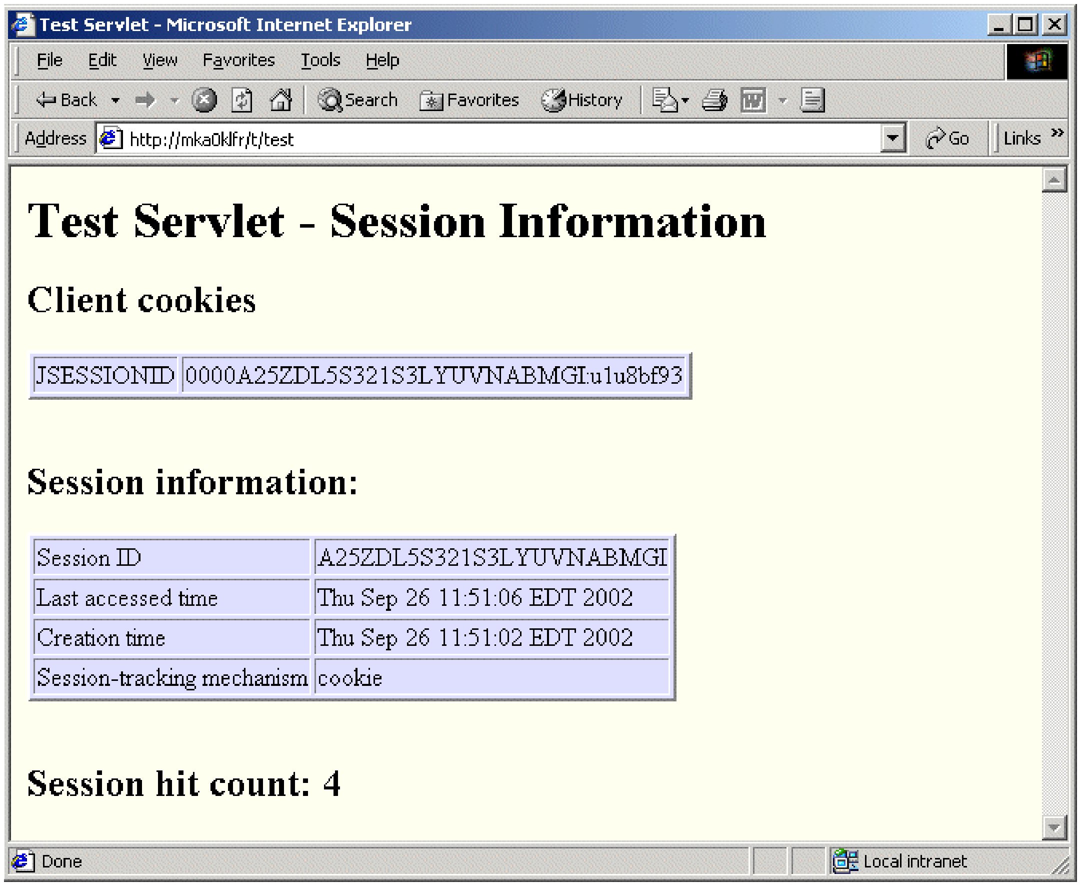
The Servlet 2.3 specification requires that an HTTP session be:

- Accessible only to the Web application that created the session. The session ID can be shared across Web applications, but not the session data.
- Handled by a single JVM for that application at any one time.

This means that in a clustered environment, any HTTP requests that are associated with an HTTP session must be routed to the same Web application in the same JVM. This ensures that all of the HTTP requests are processed with a consistent view of the user's HTTP session. The exception to this rule is when the cluster member fails or has been shut down.

WebSphere is able to assure that session affinity is maintained in the following way: Each server ID is appended to the session ID. When an HTTP session is created, its ID is passed back to the browser as part of a cookie or URL encoding. When the browser makes further requests, the cookie or URL encoding will be sent back to the Web server. The Web server plug-in examines the HTTP session ID, in the cookie or URL encoding, extracts the unique ID of the cluster member handling the session, and forwards the request.

This can be seen in Figure 9-7 on page 354, where the session ID from the HTTP header (request.getHeader("Cookie")) is displayed along with the session ID from session.getId(). The application server ID is appended to the session ID from the HTTP header. Note also that the first four characters of HTTP header session ID are the cache identifier used to determine the validity of cache entries.


Test Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://mka0klfr/t/test
Go Links
Done
Local intranet

Test Servlet - Session Information

Client cookies

JSESSIONID	0000A25ZDL5S321S3LYUVNABMGI:u1u8bf93

Session information:

Session ID	A25ZDL5S321S3LYUVNABMGI
Last accessed time	Thu Sep 26 11:51:06 EDT 2002
Creation time	Thu Sep 26 11:51:02 EDT 2002
Session-tracking mechanism	cookie

Session hit count: 4

Figure 9-7 Session ID containing the server ID and cache ID

The JSESSIONID cookie can be divided into four parts: cache ID, session ID, separator, and clone ID. Table 9-1 shows the mappings of them based on the example in Figure 9-7. A clone ID is an ID of a cluster member.

Table 9-1 Cookie mapping

content	value in the example
Cache ID	0000
Session ID	A25ZDL5S321S3LYUVNABMGI
separator	:
Clone ID	u1u8bf93

The application server ID can be seen in the Web server plug-in configuration file, plugin-cfg.xml file, as shown in Example 9-6.

Example 9-6 Server ID from plugin-cfg.xml file

```
<?xml version="1.0"?>
<Config>
......
   <ServerCluster Name="MyCluster">
      <Server CloneID="u1u8bf93" LoadBalanceWeight="2" Name="MyClusterServer1">
        <Transport Hostname="9.24.104.124" Port="9082" Protocol="http"/>
        <Transport Hostname="9.24.104.124" Port="9445" Protocol="https">
    ......
</Config>
```

Note: Session affinity can still be broken if the cluster member handling the request fails. To avoid losing session data, you can use persistent session management. In persistent sessions mode, cache ID and server ID will change in the cookie when there is a failover or when the session is read from the persistent store. So don’t rely on the value of the session cookie remaining the same for a given session.

9.8.1 Session affinity and failover

Server clusters provide a solution for failure of an application server. Sessions created by cluster members in the server cluster share a common persistent session store. Therefore, any cluster member in the server cluster has the ability to see any user’s session saved to persistent storage. If one of the cluster members fail, the user may continue to use their session information from another cluster member in the server cluster. This is known as failover. Failover works regardless of whether the nodes reside on the same machine or several machines.

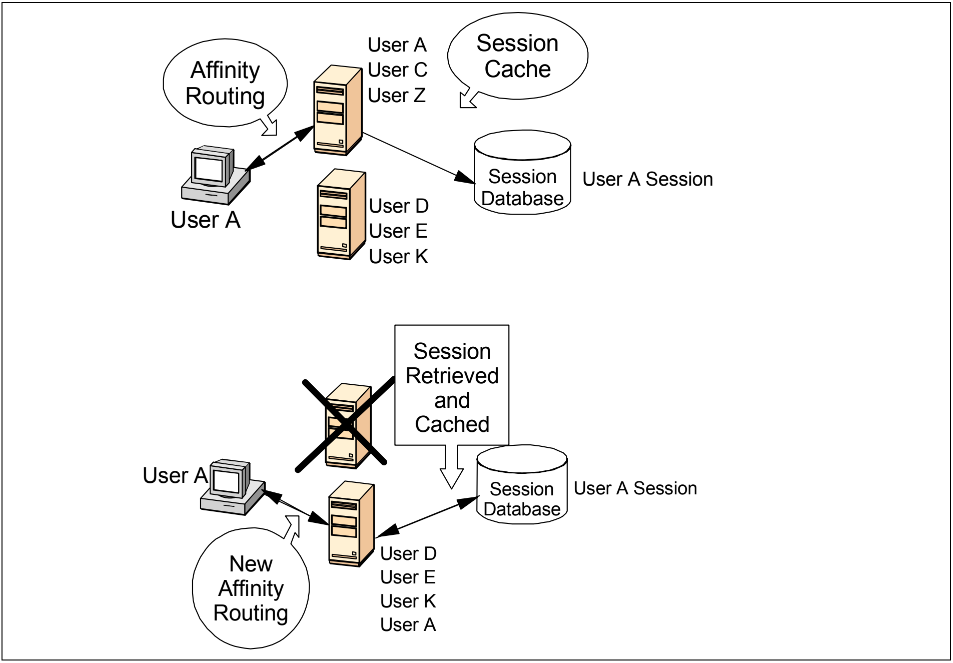


Figure 9-8 Session affinity and failover

Note: According to the Servlet 2.3 specification, only a single cluster member may control/access a given session at a time.

After a failure, WebSphere redirects the user to another cluster member, and the user's session affinity switches to this replacement cluster member. After the initial read from the persistent store, the replacement cluster member places the user's session object in the in-memory cache (assuming the cache has space available for additional entries).

The Web server plug-in maintains the cluster member list in order and will pick the cluster member next in its list to avoid the breaking of session affinity. From then on, requests for that session will go to the selected cluster member. The requests for the session will go back to the failed cluster member when the failed cluster member comes back up.

WebSphere provides session affinity on a best-effort basis. There are narrow windows where session affinity will fail. These windows are:

1. When a cluster member is recovering from a crash, there exists a window where concurrent requests for the same session could end up in different cluster members. This is because the Web server is multi-processed and

each process separately maintains its own retry timer value and list of available cluster members. The end result is that requests being processed by different processes may end up being sent to more than one cluster member after at least one process has determined that the failed cluster member is up.

To avoid or limit exposure due to this scenario, if your cluster members are expected to crash very seldom and are expected to recover fairly quickly, consider setting the retry timeout to a small value. This will narrow the window during which multiple requests being handled by different processes get routed to multiple cluster members.

2. A server overload can cause requests belonging to the same session to go to different cluster members. This can occur even if all the cluster members are up and running. For each cluster member, there is a backlog queue where an entry is made for each request sent by the Web server plug-in waiting to be picked up by a worker thread in the servlet engine. If the depth of this queue is exceeded, the Web server plug-in will start receiving responses that the cluster member is not available. This failure will be handled in the same way by the Web server plug-in as an actual JVM crash.

   Examples of when this can happen are:

   – If the servlet engine has not been set up with an appropriate number of threads to handle the user load.
   – If the servlet engine threads are taking a long time to process the requests for various reasons, such as applications taking a long time to execute, resources being used by applications taking a long time, and so on.

9.9 Persistent session management

By default, WebSphere places session objects in memory. However, the administrator has the option of enabling persistent session management, which instructs WebSphere to place session objects in a persistent store.

Administrators should enable persistent session management when:

- The user’s session data must be recovered by another cluster member after a cluster member in a cluster fails or is shut down.
- The user’s session data is too valuable to lose through unexpected failure at the WebSphere node.
- The administrator desires better control of the session cache memory footprint. By sending cache overflow to a persistent session store, the administrator controls the number of sessions allowed in memory at any given time.

There are two ways to configure session persistence in WebSphere V5. In addition to the traditional database persistence, there is a new feature that provides memory-to-memory session state replication using WebSphere internal messaging.

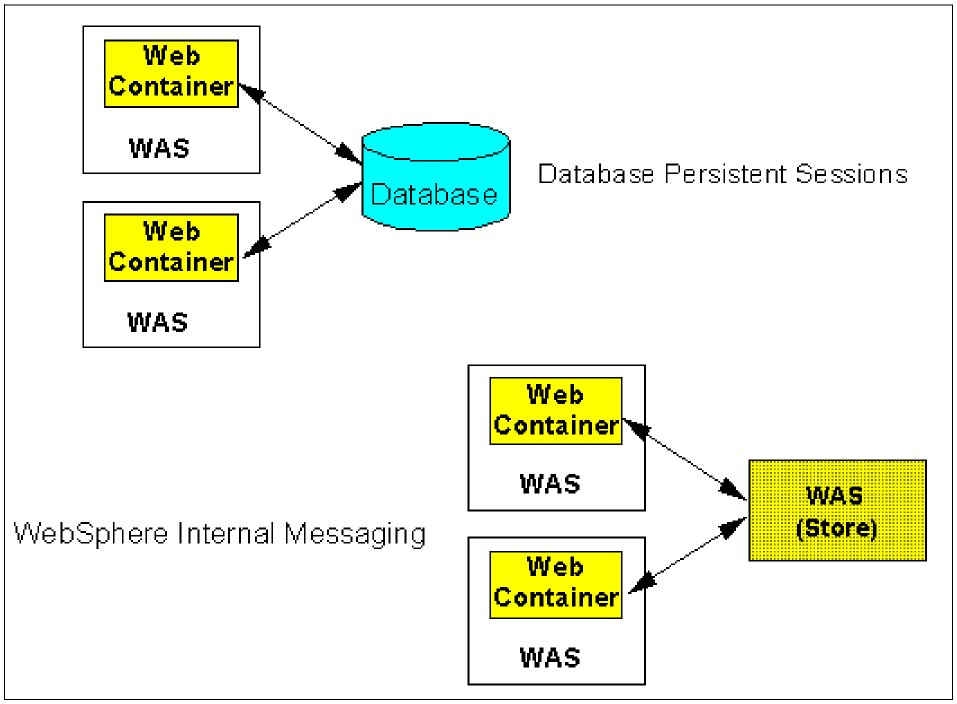


Figure 9-9 Persistent session options

All information stored in a persistent session store must be serialized. As a result, all of the objects held by a session must implement java.io.Serializable if the session needs to be stored in a persistent session store.

In general, consider making all objects held by a session serialized, even if immediate plans do not call for the use of persistent session management. If the Web site grows, and persistent session management becomes necessary, the transition between local and persistent management occurs transparently to the application if the sessions only hold serialized objects. If not, a switch to persistent session management requires coding changes to make the session contents serialized.

Persistent session management does not impact the session API, and Web applications require no API changes to support persistent session management. However, as mentioned above, applications storing un-serializable objects in their sessions require modification before switching to persistent session management.

If database persistence is used, using multi-row sessions becomes important if the size of the session object exceeds the size for a row, as permitted by the WebSphere session manager. If the administrator requests multi-row session support, the WebSphere session manager breaks the session data across multiple rows as needed. This allows WebSphere to support large session objects. Also, this provides a more efficient mechanism for storing and retrieving session contents under certain circumstances. See 9.9.6, “Single versus multi-row schemas (database persistence)” on page 381 for information on this feature.

Using a cache lets the session manager maintain a cache of most recently used sessions in memory. Retrieving a user session from the cache eliminates a more expensive retrieval from the persistent store. The session manager uses a “least recently used” scheme for removing objects from the cache. Session data is stored to the persistent store based on the write frequency and write option selected.

9.9.1 Enabling database persistence

It is assumed in this section that the following tasks have already completed before enabling database persistence:

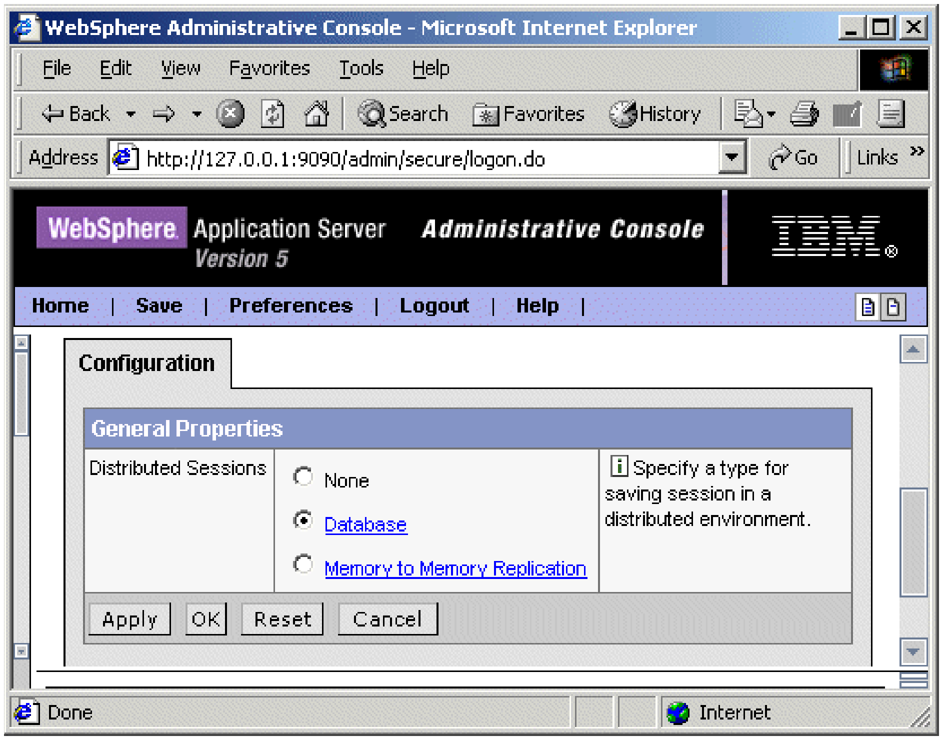
1. Create a session database. In this example, it is assumed that the data source JNDI name is jdbc/Sessions.
2. Create a JDBC provider and data source. See 12.1, “JDBC resources” on page 550 and 12.2, “Creating a data source” on page 555.

Note: The following example illustrates the steps to enable database persistence at the application server level. In WebSphere V5, session management settings can also be performed at the enterprise application level and the Web application level.

In order to enable database persistence, repeat the following steps for each application server.

1. Click **Servers -> Application Servers**
2. Select the server.
3. Click **Web Container** in the Additional Properties table.

4. Click **Session Management**
5. Click **Distributed Environment Settings**
6. Select **Database** and click **Database**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do Go Links
WebSphere Application Server Version 5 Administrative Console IBM
Home | Save | Preferences | Logout | Help |
Done Internet

Configuration

General Properties

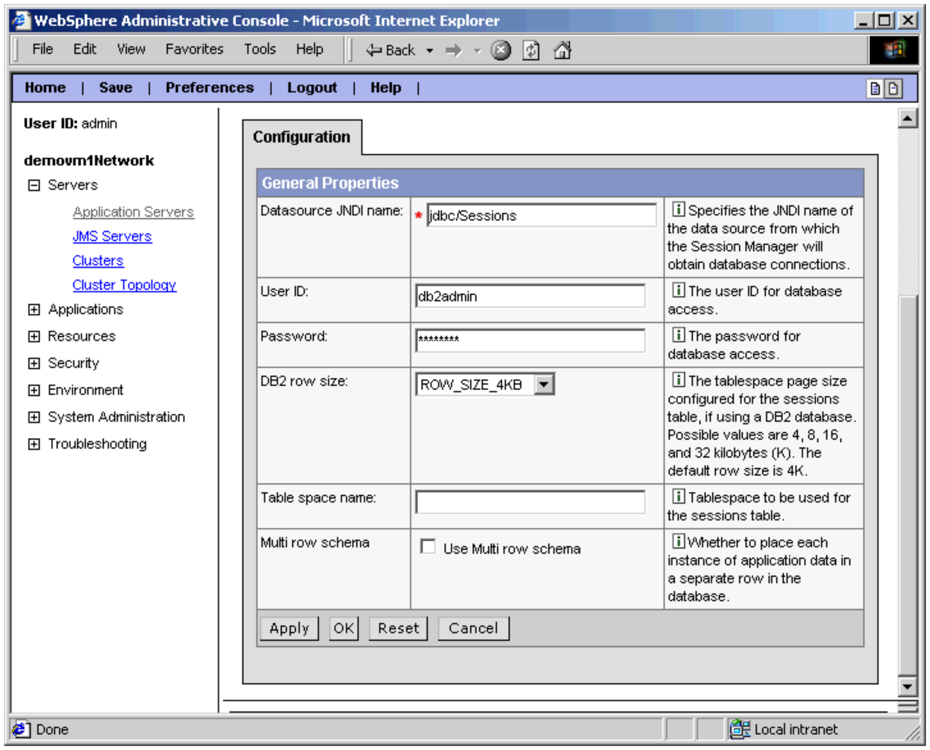
Distributed Sessions	○ None ● Database ○ Memory to Memory Replication	Specify a type for saving session in a distributed environment.

Apply OK Reset Cancel

Figure 9-10 Distributed Environment Setting (database)

7. Enter the database information:
   a. Enter the data source JNDI name. The data source must be a non-JTA enabled data source.
   b. Enter the user ID and password to be used to access the database.
   c. If you are using DB2 and you anticipate requiring row sizes greater than 4 KB, select the appropriate value from the DB2 row size pull-down. See 9.9.5, “Using larger DB2 page sizes (database persistence)” on page 380 for more information.
   d. If DB2 row size is other than 4 KB, you are required to enter the name of tablespace. See “Using larger DB2 page sizes (database persistence)” on page 380.

e. If you intend to use a multi-row schema, select **Use Multi row schema**. See 9.9.6, “Single versus multi-row schemas (database persistence)” on page 381 for more information.


WebSphere Administrative Console - Microsoft Internet Explorer

User ID: admin

demovm1Network

Configuration

General Properties

Datasource JNDI name:	* jdbc/Sessions	Specifies the JNDI name of the data source from which the Session Manager will obtain database connections.
User ID:	db2admin	The user ID for database access.
Password:	********	The password for database access.
DB2 row size:	ROW_SIZE_4KB	The tablespace page size configured for the sessions table, if using a DB2 database. Possible values are 4, 8, 16, and 32 kilobytes (K). The default row size is 4K.
Table space name:		Tablespace to be used for the sessions table.
Multi row schema	☐ Use Multi row schema	Whether to place each instance of application data in a separate row in the database.

Figure 9-11 Database

8. Click **OK**.

After you have updated each server, save the configuration changes, synchronize them with the servers, and restart the application servers.

9.9.2 WebSphere internal messaging

WebSphere internal messaging is a new feature of WebSphere Application Server V5 that enables sessions to be shared among application servers without using a database. Using this method, sessions are stored in the memory of an application server, providing the same functionality as a database for session persistence. Separate threads handle this functionality within an existing application server process.

Note: The internal messaging system is not associated with the embedded WebSphere JMS provider. Instead the necessary code is provided as part of the core WebSphere Application Server libraries. As such, the JMS Server does not need to be started.

The advantages of using this method of session persistence are:

- Flexible configuration options such as peer-peer and client/server.
- Eliminates the overhead and cost of setting up and maintaining a real-time production database.
- Eliminates the single point of failure problem that can occur when using a database.
- Better performance than using a database.
- Session information being transferred between application servers can be encrypted.
- Partitions can be set up to allow application servers to listen only on configured channels. Partitions reduce overhead of each application server because not all application servers need to store all sessions in the domain.

The WebSphere internal messaging can be set up as peer-to-peer (the default) or client/server.

Peer-to-peer topology

Figure 9-12 on page 363 shows an example of peer-to-peer topology. Each application server stores sessions in its own memory. It also stores sessions to and retrieves session from other application servers. In other words, each application server acts as a client by retrieving sessions from other application servers, and each application server acts as a server by providing sessions to other application servers.

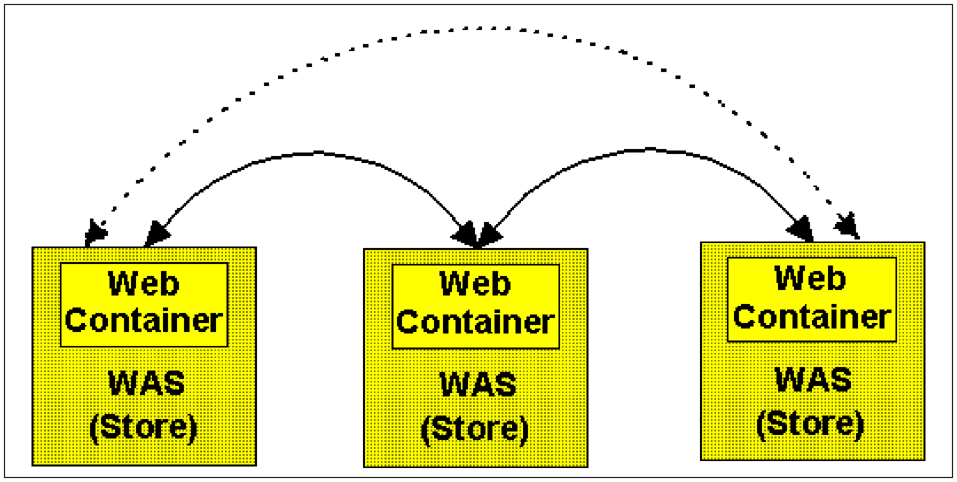


Figure 9-12 Example of peer-to-peer topology

The advantage of this topology is that no additional processes and/or products are required to avoid a single point of failure. This reduces the time and cost required to configure and maintain additional processes and/or products.

One of the disadvantages of this topology is that it can consume large amounts of memory in networks with many users, since each server has a copy of all sessions. For example, assuming that a single session consumes 10 KB and one million users have logged into the system, each application server consumes 10 GB of memory in order to keep all sessions in its own memory. Another disadvantage is that every change to a session must be replicated to all application servers. This can cause a performance impact.

Client/server topology

Figure 9-13 on page 364 shows an example of client/server topology. In this setup, application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that just store sessions but do not respond to the users' requests. Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact.

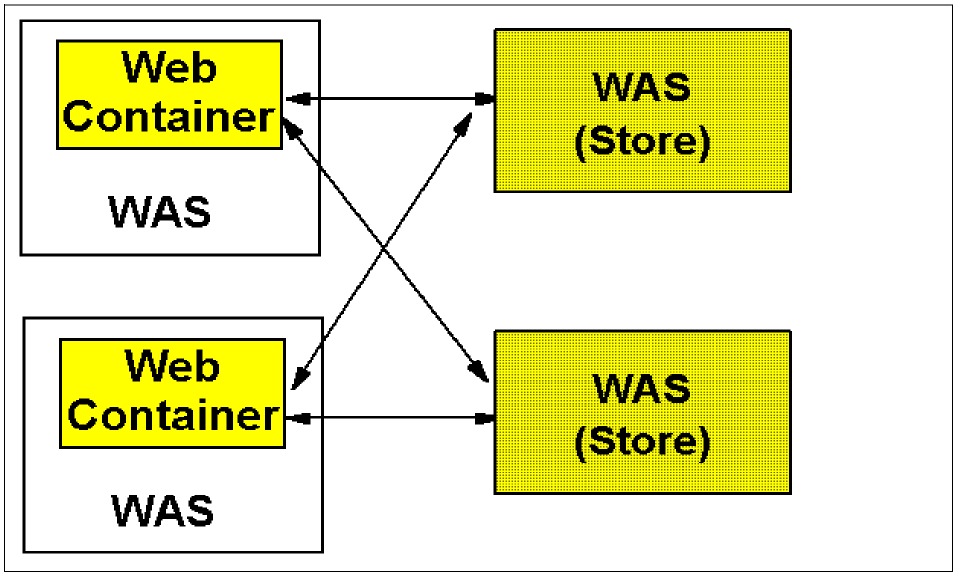


Figure 9-13 Example of client/server topology

The advantage of this topology is that it clearly distinguishes the role of client and server. Only replication servers keep all sessions in their memory and only the clients interact with users. This reduces the consumption of memory on each application server and reduces the performance impact, since session information is only sent to the servers.

One of the disadvantages of this topology is that additional application servers have to be configured and maintained over and above those that interact with users. It is recommended that you have multiple replication servers configured in order to avoid a single point of failure.

Replication domain and replicator entry

WebSphere internal messaging uses a small and fast publish/subscribe engine supplied with WebSphere. It consists of two key elements: replication domain and replication entry. These are defined at the cell level.

A replication domain defines the set of replicator processes that communicate with each other and share a common publish/subscribe cluster. Multiple replication domains can reside in a cell.

A replicator runs inside an application server process. The replicator entry defines its basic settings, such as a communication port among replicators and a communication port with clients. A replication domain can contain multiple replicator entries.

Enabling WebSphere internal messaging

It is assumed in this section that the following tasks have already completed before enabling WebSphere internal messaging:

1. You have created a cluster consisting of at least two application servers.

   In this example, we are working with a cluster called MyCluster. It has two servers, MyClusterServer1 and MyClusterServer2.

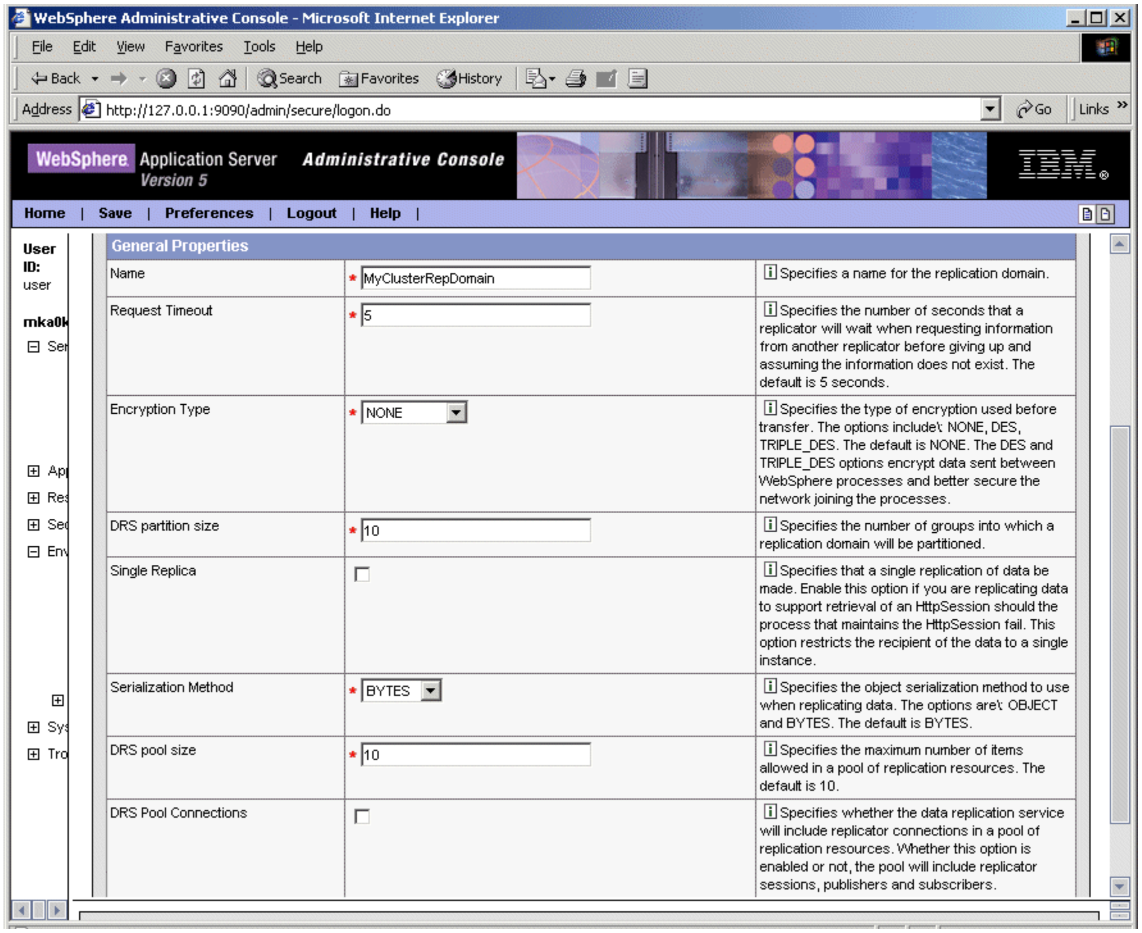
2. You have installed applications to the cluster.
3. The Web server plug-in has been generated and copied to the Web server.

Note: This example illustrates setting up the replication domain and replicators after the cluster has been created. You also have the option of creating the replication domain and the replicator for the first server in the cluster when you create the cluster.

In order to enable WebSphere internal messaging, you need to do the following:

1. Create a replication domain to define the set of replicator processes that communicate with each other.

   a. Select **Environment -> Internal Replication Domains.** Click **New**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: user
mka0k
Ser
App
Res
Sec
Env
Sys
Tro

General Properties

Name	* MyClusterRepDomain	Specifies a name for the replication domain.
Request Timeout	* 5	Specifies the number of seconds that a replicator will wait when requesting information from another replicator before giving up and assuming the information does not exist. The default is 5 seconds.
Encryption Type	* NONE	Specifies the type of encryption used before transfer. The options include: NONE, DES, TRIPLE_DES. The default is NONE. The DES and TRIPLE_DES options encrypt data sent between WebSphere processes and better secure the network joining the processes.
DRS partition size	* 10	Specifies the number of groups into which a replication domain will be partitioned.
Single Replica	☐	Specifies that a single replication of data be made. Enable this option if you are replicating data to support retrieval of an HttpSession should the process that maintains the HttpSession fail. This option restricts the recipient of the data to a single instance.
Serialization Method	* BYTES	Specifies the object serialization method to use when replicating data. The options are: OBJECT and BYTES. The default is BYTES.
DRS pool size	* 10	Specifies the maximum number of items allowed in a pool of replication resources. The default is 10.
DRS Pool Connections	☐	Specifies whether the data replication service will include replicator connections in a pool of replication resources. Whether this option is enabled or not, the pool will include replicator sessions, publishers and subscribers.

Figure 9-14 Create a replication domain

At a minimum, you need to enter a name for the replication domain. The name must be unique within the cell. In this example, MyClusterRepDomain is used as name and defaults are used as other properties.

Encrypted transmission achieves better security but can impact performance. If DES or TRIPLE_DES is specified, a key for data transmission is generated. It is recommended that you generate a key by clicking the **RegenerateKey** button periodically to enhance security.

The DRS (Data Replication Service) partition size specifies the number of partitions in a replication domain. This setting determines the number of partitions in the replication domain. Later, as you define the replicators in the domain, each replicator is configured to listen to partitions by ID (see Figure 9-17 on page 370).

The serialization method specifies the method of serialization for replicating data. Session data can be replicated as an object or as bytes.

- Using object replication is generally faster than byte replication, but it requires that class definitions be available on the receiving side. This means that all class types of the objects in the session must be defined in the classpath of all application servers that use and/or store the session.
- Storing session data as byte data is convenient because there is no classpath definition required on the receiving side. This option is useful in the client/server replication topology where the session data is stored but does not need to be instantiated.

b. Click **Apply**.

2. Add replicator entries by repeating the following steps for each application server in the cluster:

   a. Click **Replicator Entries** in the Additional Properties table for the new replication domain.

   b. Click **New**.

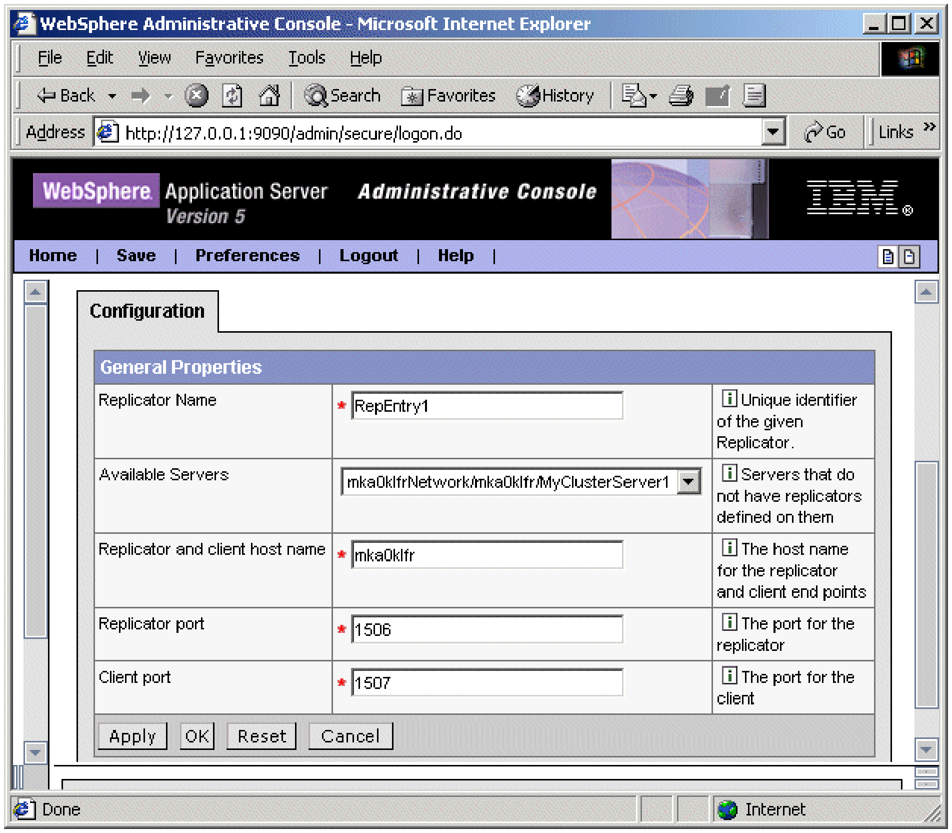
   The following values are used for the first replication entry in this example:

   - Name: `RepEntry1`
   - Server: `m10df51fNetwork/m10df51f/MyClusterServer1.`

     You can select from the application servers that do not already have a replicator defined.
   - Replicator and client host name (application server host): `mka0klfr`
   - Replicator port: `1506.`

     The replicator ports are used for replicator-to-replicator communication. The port must be unique to the server.
   - Client port: `1507.`

     The client port is used for application server-to-replicator communication. The port must be unique to the host.


WebSphere Administrative Console - Microsoft Internet Explorer

Address http://127.0.0.1:9090/admin/secure/logon.do

Configuration

General Properties

Replicator Name	* RepEntry1	Unique identifier of the given Replicator.
Available Servers	mka0klfrNetwork/mka0klfr/MyClusterServer1	Servers that do not have replicators defined on them
Replicator and client host name	* mka0klfr	The host name for the replicator and client end points
Replicator port	* 1506	The port for the replicator
Client port	* 1507	The port for the client

WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Apply
OK
Reset
Cancel
Done
Internet

Figure 9-15 Adding a replicator entry

c. Click **OK.**

The following values are used for the second replication entry in this example:

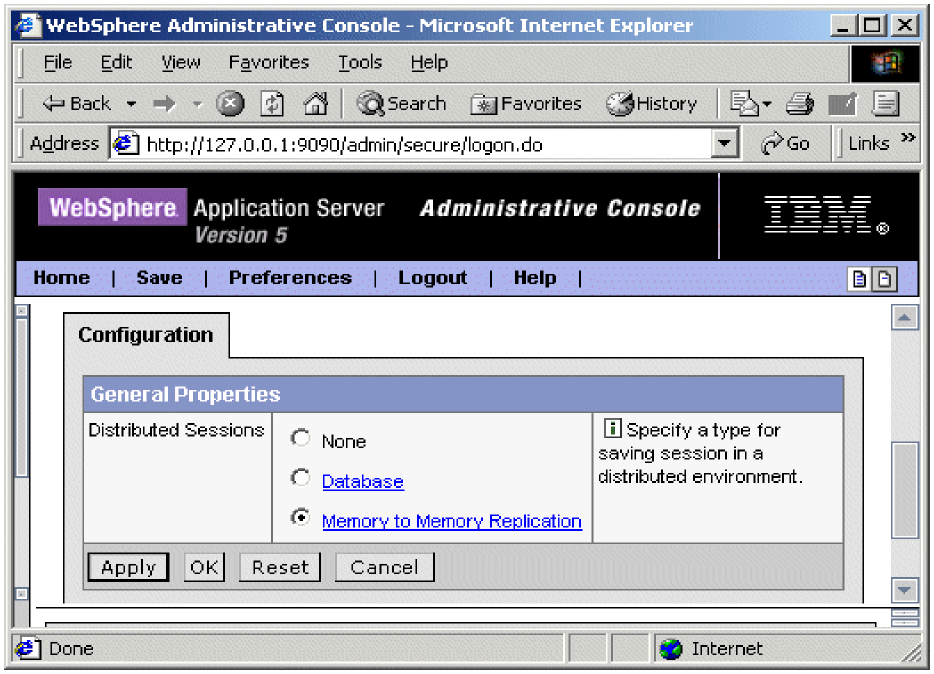
- Name: `RepEntry2`
- Server: `m10df51fNetwork/m10df51f/MyClusterServer2`
- Replicator and client host name: `mka0klfr`
- Replicator port: `1508`
- Client port: `1509`

d. Save the configuration changes.

3. Configure cluster members.

   Repeat the following steps for each application server:

   a. Select **Servers -> Application Servers**
   b. Click the application server name. In this example, **MyClusterServer1** and **MyClusterServer2** are selected as application servers respectively.
   c. Click **Web Container** in the Additional Properties table.
   d. Click **Session Management**.
   e. Click **Distributed Environment Settings**.
   f. Select **Memory to Memory Replication**.

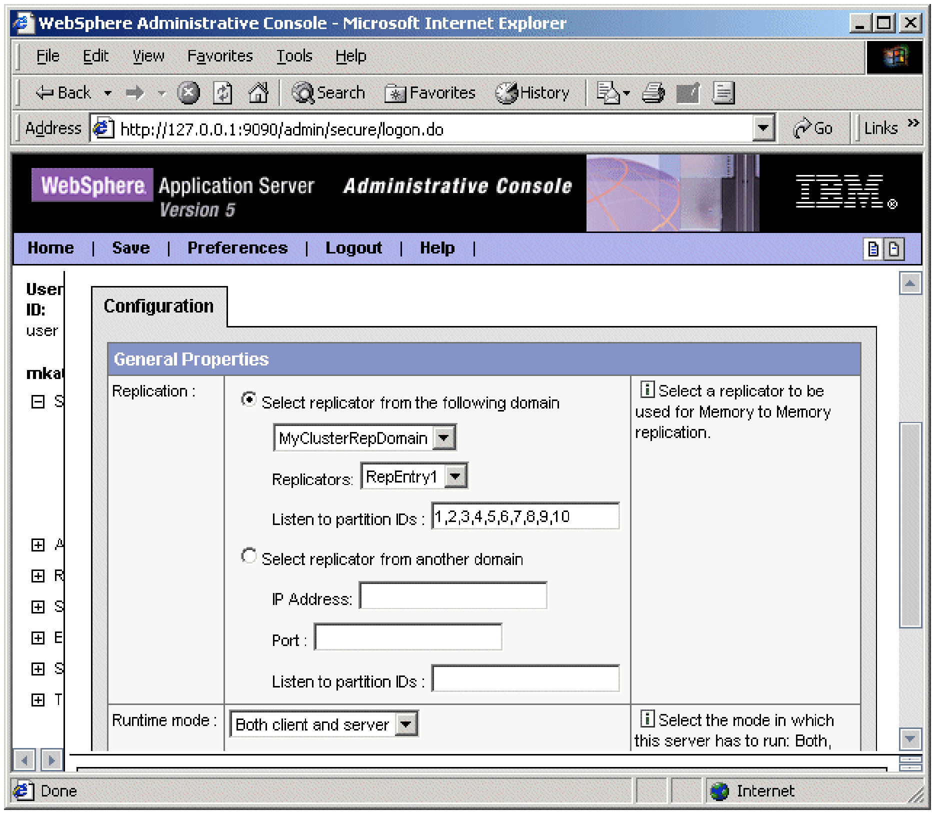

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Configuration
Apply OK Reset Cancel
Done
Internet

General Properties

Distributed Sessions	None Database [✓ Memory to Memory Replication]	Specify a type for saving session in a distributed environment.

Figure 9-16 Distributed environment settings

g. Choose a replicator domain and replicator either from listed domains or from another domains.


General Properties

Replication :
- [✓] Select replicator from the following domain: MyClusterRepDomain
  - Replicators: RepEntry1
  - Listen to partition IDs : 1,2,3,4,5,6,7,8,9,10
- [ ] Select replicator from another domain
  - IP Address:
  - Port :
  - Listen to partition IDs :

Select a replicator to be used for Memory to Memory replication.

Runtime mode : Both client and server

Select the mode in which this server has to run: Both,

Figure 9-17 Internal messaging settings

Select the replication topology by specifying the runtime mode. Selecting **Both client and server** identifies this as a peer-to-peer topology. In a client/server topology, select **Client only** for application servers that will be responding to user requests. Select **Server only** for those that will be used as replication servers.

In this example, we specified the following values for the first application server:

- Replicator Domain: `MyClusterRepDomain`
- Replicator: `RepEntry1`

The values chosen for the second application server were:

- Replicator Domain: `MyClusterRepDomain`
- Replicator: `RepEntry2`

h. Click **OK**.

4. Save the configuration and restart the cluster. You can restart the cluster by selecting **Servers -> Clusters**. Check the cluster, and click **Stop**. After the messages indicate the cluster has stopped, click **Start**.

Configuration file results

The replication domain configuration is written to the multibroker.xml.

Example 9-7 Replication domain configuration in multibroker.xml

```
<multibroker:MultibrokerDomain xmi:id="MultibrokerDomain_1" name="RepDomain1">
       <defaultDataReplicationSettings xmi:id="DataReplication_1"
   requestTimeout="5" encryptionType="NONE" encryptionKeyValue="">
      <partition xmi:id="DRSPartition_1" size="10" partitionOnEntry="false"/>
     <serialization xmi:id="DRSSerialization_1" entrySerializationKind="BYTES"
propertySerializationKind="BYTES"/>
      <pooling xmi:id="DRSConnectionPool_1" size="10" poolConnections="false"/>
</defaultDataReplicationSettings>
  </multibroker:MultibrokerDomain>
```

Each replication entry is written to the multibroker.xml configuration file and the associated application server's server.xml. In multibroker.xml, the replication entries are added under the chosen replication domain's stanza.

Example 9-8 Replication entry configuration in multibroker.xml

```
<multibroker:MultibrokerDomain xmi:id="MultibrokerDomain_1" name="RepDomain1">
    <entries xmi:id="MultiBrokerRoutingEntry_1" brokerName="RepEntry1">
      <brokerEndPoint xmi:id="EndPoint_1" host="pentiv1" port="20000"/>
      <clientEndPoint xmi:id="EndPoint_2" host="pentiv1" port="20001"/>
    </entries>
    <defaultDataReplicationSettings xmi:id="DataReplication_1"
requestTimeout="5" encryptionType="NONE" encryptionKeyValue="">
      <partition xmi:id="DRSPartition_1" size="10" partitionOnEntry="false"/>
     <serialization xmi:id="DRSSerialization_1" entrySerializationKind="BYTES"
propertySerializationKind="BYTES"/>
      <pooling xmi:id="DRSConnectionPool_1" size="10" poolConnections="false"/>
    </defaultDataReplicationSettings>
  </multibroker:MultibrokerDomain>
```

In server.xml, it is listed as a new component/service to run in the application server runtime.

Example 9-9 Replication entry configuration in server.xml

```
<components xmi:type="datareplicationserver:SystemMessageServer"
xmi:id="SystemMessageServer_1" brokerName="RepEntry1" enable="true"
domainName="RepDomain1"/>
```

Configuring an application server to use a replication domain for session persistence updates the server.xml.

Example 9-10 Server.xml updates

```
<sessionDRSPersistence xmi:id="DRSSettings_1" dataReplicationMode="BOTH"
messageBrokerDomainName="RepDomain1" preferredLocalDRSBrokerName="RepEntry1"/>
```

When an application server is started, the server.xml file is read, which indicates that the DRS component needs to be loaded and run in this server's runtime. The ports and mode used by this server's DRS client and broker ports are read from the mutlibroker.xml. When the session manager in the server acts as a DRS client, it uses the port listed in the application server's stanza in serverindex.xml.

Table 9-2 DRS port use

IP port	Configuration file	Usage
client DRS	serverindex.xml	Used by the client to receive session data from the replication server (in client/server mode)
client	multibroker.xml	The client port enables communication between an application server process and a replicator. It provides communication between the client port and the application server. Runs as a listener on TCP/IP port.
replicator	multibroker.xml	The replicator port enables communication among replicators. It provides replicator port-to-replicator communication. Runs as a listener on TCP/IP port.

When the application server is running and it has the DRS component/service listed in its server.xml, then the broker and client ports of the associated replication entry (read from multibroker.xml) will have TCP/IP listeners running. Application servers with no DRS configured for its session manager will not have this service running, or have listeners on DRS ports.

Operationally, the start/stop of each application server starts/stops the associated DRS replication endpoint/entry. The DRS can't be started/stopped all together, because each application server process runs its own DRS service.

Each DRS service knows about the state of the others based upon connectivity information it can read from the configuration files. This is a reason why the server's DRS ports are located in the cell-wide multibroker.xml. If they were stored in each server's server.xml, servers on other nodes would not have access to that file and hence the host and port info.

9.9.3 Session management tuning

Performance tuning for session management consists of defining the following:

- How often session data is written (write frequency settings).
- How much data is written (write contents settings).
- When the invalid sessions are cleaned up (session cleanup settings).

Writing frequency settings

You can select from three different settings that determine how often session data is written to the persistent data store:

- End of servlet service: If the session data has changed, it will be written to the persistent store after the servlet finishes processing an HTTP request.
- Manual update: The session data will be written to the persistent store when the sync() method is called on the IBMSession object.
- Time-based: The session data will be written to the persistent store based on the specified write interval value.

Note: The last access time attribute is always updated each time the session is accessed by the servlet or JSP whether the session is changed or not. This is done to make sure the session does not time out.

- If you choose the end of servlet service option, each servlet or JSP access will result in a corresponding persistent store update of the last access time.
- If you select the manual update option, the update of the last access time in persistent store occurs on sync() call or at later time.
- If you use time-based updates, the changes are accumulated and written in a single transaction. This can significantly reduce the amount of I/O to the persistent store.

See “Reducing persistent store I/O” on page 394 for options available to change this database update behavior.

Let's consider an example where the Web browser accesses the application once every five seconds:

- In End of servlet service mode, the session would get written out every five seconds.
- In Manual update mode, the session gets written out whenever the servlet issues IBMSession.sync(). It is the responsibility of the servlet writer to use the IBMSession interface instead of the HttpSession Interface and the servlets/JSPs must be updated to issue the sync().
- In Time-based mode, the servlet or JSP need not use the IBMSession class nor issue IBMSession.sync(). If the write interval is set to `120` seconds, then the session data gets written out at most every 120 seconds.

End of servlet service

When the write frequency is set to the end of servlet service option, WebSphere writes the session data to the persistent store at the completion of the HttpServlet.service() method call. Exactly what is written depends on the write content settings.

Manual update

In manual update mode, the session manager only sends changes to the persistent data store if the application explicitly requests a save of the session information.

Note: Manual updates use an IBM extension to HttpSession that is not part of the Servlet 2.3 API.

Manual update mode requires that an application developer use the IBMSession class for managing sessions. When the application invokes the sync() method, the session manager writes the modified session data and last access time to the persistent store. The session data that is written out to the persistent store is controlled by the write contents option selected.

If the servlet or JSP terminates without invoking the sync() method, the session manager saves the contents of the session object into the session cache (if caching is enabled), but does not update the modified session data in the session database. The session manager will only update the last access time in the persistent store asynchronously, at later time.

Example 9-11 shows how the IBMSession class can be used to manually update the persistent store.

Example 9-11 Using IBMSession for manual update of the persistent store

```
public void service (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
{
   // Use the IBMSession to hold the session information
   // We need the IBMSession object because it has the manual update
   // method sync()
   com.ibm.websphere.servlet.session.IBMSession session =
      (com.ibm.websphere.servlet.session.IBMSession)req.getSession(true);

   Integer value = 1;

   //Update the in-memory session stored in the cache
   session.putValue("MyManualCount.COUNTER", value);

   //The servlet saves the session to the persistent store
   session.sync();
}
```

This interface gives the Web application developer additional control of when (and if) session objects go to the persistent data store. If the application does not invoke the sync() method, and manual update mode is specified, the session updates goes only to the local session cache, not the persistent data store. Web developers use this interface to reduce unnecessary writes to the session database, and thereby to improve overall application performance.

All servlets in the Web application server must perform their own session management in manual update mode.

Note: In WebSphere V4, manual update can be performed when the manual update option is selected. In WebSphere V5, invoking sync() method always saves the session to the persistent store.

Time-based writes to the session database

Using the time-based write option will write session data to the persistent store at a defined write interval. The reasons for implementing time-based write lies in the changes introduced with the Servlet 2.2 API. The Servlet 2.2 specification introduced two key concepts:

- It limits the scope of a session to a single Web application.
- It both explicitly prohibits concurrent access to an HttpSession from separate Web applications but allows for concurrent access within a given JVM.

Because of these changes, WebSphere provides the session affinity mechanism that assures us that an HTTP request is routed to the Web application handling its HttpSession. This assurance still holds in a WLM environment when using persistent HttpSessions. This means that the necessity to immediately write the session data to the persistent store can now be relaxed somewhat in these environments (as well as non-clustered environments), since the persistent store is now really only used for failover and session cache full scenarios.

With this in mind, it is now possible to gain potential performance improvements by reducing the frequency of persistent store writes.

Note: Time-based writes requires session affinity for session data integrity.

The following details apply to time-based writes:

- The expiration of the write interval does not necessitate a write to the persistent store unless the session has been touched (that is, getAttribute/setAttribute/removeAttribute was called) since the last write.
- If a session write interval has expired and the session has only been retrieved (that is, request.getSession() was called since the last write) then the last access time will be written to the persistent store regardless of the write contents setting.
- If a session write interval has expired and the session properties have been either accessed or modified since the last write then the session properties will be written out in addition to the last access time. Which session properties get written out is dependent on the write contents settings.
- Time-based write allows the servlet or JSP to issue IBMSession.sync() to force the write of session data to the database.
- If the time between session servlet requests (for a particular session) is greater than the write interval, then the session effectively gets written out after each service method invocation.
- The session cache should be large enough to hold all of the active sessions. Failure to do this will result in extra persistent store writes, since the receipt of a new session request may result in writing out the oldest cached session to the persistent store. Or to put it another way, if the session manager has to remove the least recently used HttpSession from the cache during a full cache scenario, the session manager will write out that HttpSession (per the Write contents settings) upon removal from the cache.
- The session invalidation time must be at least twice the write interval to ensure that a session does not inadvertently get invalidated prior to getting written to the persistent store.

- A newly created session will always get written to the persistent store at the end of the service method.

Writing content settings

These options control what is written. Please refer to 9.9.7, “What is written to the persistent session database” on page 383 before selecting one of the options, since there are several factors to take into account. The options available are:

- Only update attributes: Only updated attributes are written to the persistent store.
- All session attributes: All attribute are written to the persistent store.

Session cleanup settings

WebSphere allows the administrator to defer to off hours the clearing of invalidated sessions (sessions that are no longer in use and timed out) from the persistent store. For more information, see 9.10, “Invalidating sessions” on page 387. This can be done either once or twice a day. The fields available are:

- First time of day (0-23): The first hour during which the invalidated persistent sessions will be cleared from the persistent store. This value must be a positive integer between 0 and 23.
- Second time of day (0-23): The second hour during which the invalidated persistent sessions will be cleared from the persistent store. This value must be a positive integer between 0 and 23.

The property, **Specify distributed sessions cleanup schedule**, is required to be selected to enable this option.

Also consider using schedule invalidation for intranet-style applications that have a somewhat fixed number of users wanting the same HTTP session for the whole business day.

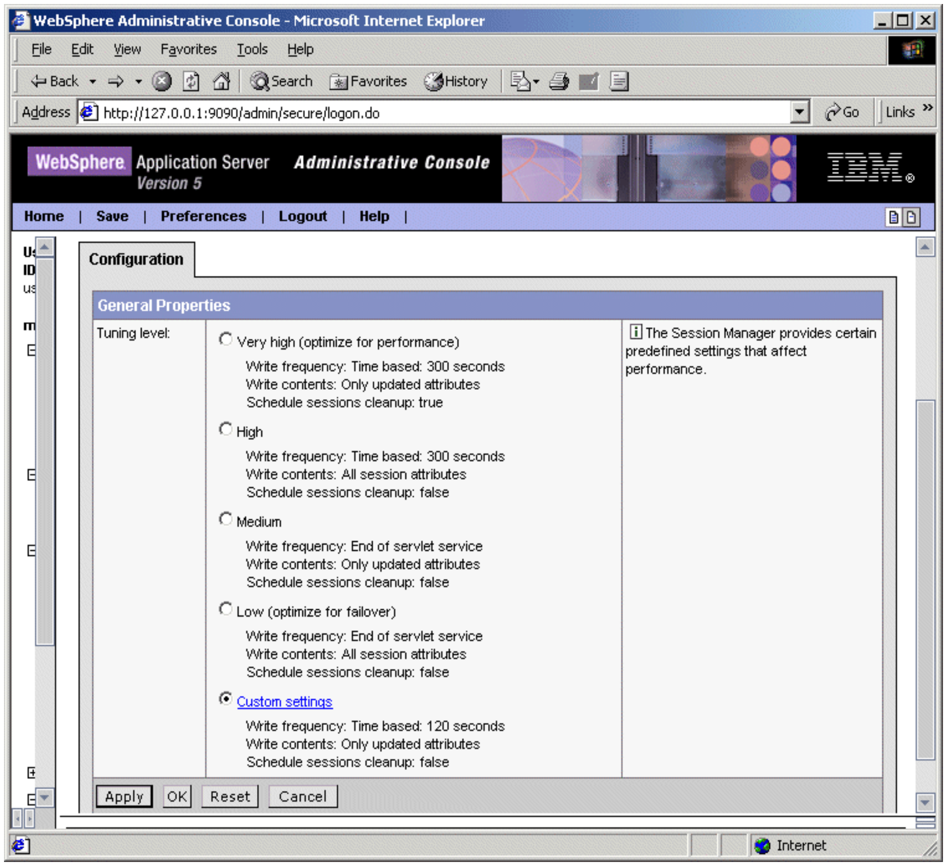
Configuration

The session management tuning parameters can be set by selecting a pre-defined tuning level or by specifically specifying each parameter. To specify the performance settings for session management:

1. Select **Servers -> Application Servers** and click the application server.

   Note: Remember, session management options can also be set at the enterprise application level (see “Application session management properties” on page 338) or at the Web module level (see “Web module session management properties” on page 338).

2. Click **Web Container**.
3. Click **Session Management**.
4. Click **Distributed Environment Settings**.
5. Select from the predefined tuning levels or click **Custom Tuning Parameters**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Configuration
General Properties
Apply OK Reset Cancel
Internet

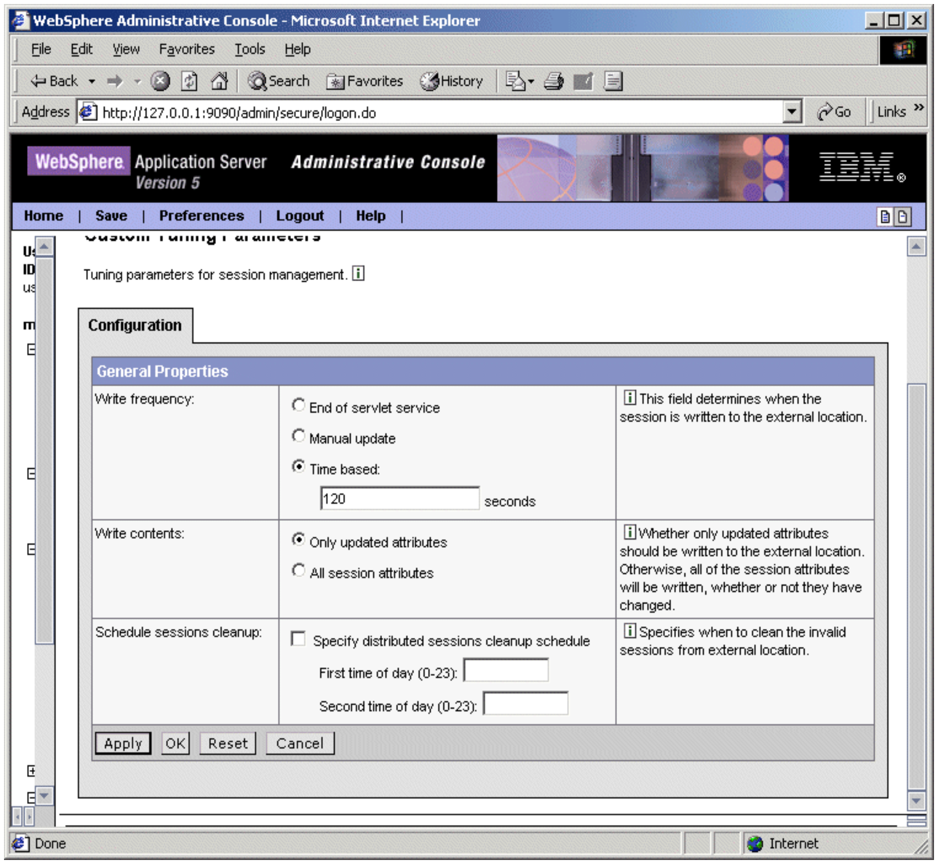
Tuning level:

- ◯ Very high (optimize for performance)
  Write frequency: Time based: 300 seconds
  Write contents: Only updated attributes
  Schedule sessions cleanup: true
- ◯ High
  Write frequency: Time based: 300 seconds
  Write contents: All session attributes
  Schedule sessions cleanup: false
- ◯ Medium
  Write frequency: End of servlet service
  Write contents: Only updated attributes
  Schedule sessions cleanup: false
- ◯ Low (optimize for failover)
  Write frequency: End of servlet service
  Write contents: All session attributes
  Schedule sessions cleanup: false
- [✓ Custom settings]
  Write frequency: Time based: 120 seconds
  Write contents: Only updated attributes
  Schedule sessions cleanup: false

The Session Manager provides certain predefined settings that affect performance.

Figure 9-18 Session management tuning parameters

If you want to set each tuning parameter explicitly, select **Custom Settings**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Done
Internet

Tuning parameters for session management.

Configuration

General Properties

Write frequency:	( ) End of servlet service ( ) Manual update (•) Time based: 120 seconds	This field determines when the session is written to the external location.
Write contents:	(•) Only updated attributes ( ) All session attributes	Whether only updated attributes should be written to the external location. Otherwise, all of the session attributes will be written, whether or not they have changed.
Schedule sessions cleanup:	[ ] Specify distributed sessions cleanup schedule First time of day (0-23): Second time of day (0-23):	Specifies when to clean the invalid sessions from external location.

Apply OK Reset Cancel

Figure 9-19 Session management tuning parameters

9.9.4 Persistent sessions and non-serializable J2EE objects

In order for the WebSphere session manager to persist a session to the persistent store, all of the Java objects in an HttpSession must be serializable (that is, implement the java.io.Serializable interface). The HttpSession can also contain the followng J2EE objects, which are not serializable:

- javax.ejb.EJBObject
- javax.ejb.EJBHome
- javax.naming.Context
- javax.transaction.UserTransaction

The WebSphere session manager works around the problem of serializing these objects in the following manner:

- EJBObject and EJBHome each have Handle and HomeHandle object attributes that are serializable and can be used to reconstruct the EJBObject and EJBHome.
- Context is constructed with a hash table based environment, which is serializable. WebSphere will retrieve the environment, then wrapper it with an internal, serializable object, so that on re-entry it can check the object type and reconstruct the Context.
- UserTransaction has no serializable attributes. WebSphere provides two options:
  a. The Web developer can place the object in the HttpSession but WebSphere will not persist it outside the JVM.
  b. WebSphere has a new public wrapper object, com.ibm.websphere.servlet.session.UserTransactionWrapper, which is serializable and requires the InitialContext used to construct the UserTransaction. This will be persisted outside the JVM and be used to reconstruct the UserTransaction.

Note: As per J2EE, a Web component may only start a transaction in a service method. A transaction that is started by a servlet or JSP must be completed before the service method returns. That is, transactions may not span Web requests from a client. If there is an active transaction after returning from the service method, WebSphere will detect it and will abort the transaction.

In general, Web developers should consider making all other Java objects held by HttpSession serializable, even if immediate plans do not call for the use of persistent session management. If the Web site grows, and persistent session management becomes necessary, the transition between local and persistent management occurs transparently to the application if the sessions hold only serializable objects. If not, a switch to persistent session management requires coding changes to make the session contents serializable.

9.9.5 Using larger DB2 page sizes (database persistence)

WebSphere supports 4 KB, 8 KB, 16 KB, and 32 KB page sizes, and hence can have larger varchar for bit data columns of ~7 KB, 15 KB, or 31 KB. Using this performance feature, we see faster persistence for HttpSession of sizes of 7 KB to 31 KB in the single-row case, or attribute sizes of 4 KB to 31 KB in the multi-row case.

Enabling this feature involves dropping any existing table created with a 4 KB buffer pool and tablespace. This also applies if you subsequently change between 4 KB, 8 KB, 16 KB, or 32 KB.

To use a page size other than the default (4 KB), do the following:

1. If the SESSIONS table already exists, drop it from the DB2 database:

   ```
   DB2 connect to session
   DB2 drop table sessions
   ```

2. Create a new DB2 buffer pool and tablespace, specifying the same page size (8 KB, 16 KB or 32 KB) for both, and assign the new buffer pool to this tablespace. The following are simple steps for creating an 8 KB page:

   ```
   DB2 connect to session
   DB2 CREATE BUFFERPOOL sessionBP SIZE 1000 PAGESIZE 8K
   DB2 connect reset
   DB2 connect to session
   DB2 CREATE TABLESPACE sessionTS PAGESIZE 8K MANAGED BY SYSTEM USING
   ('D:\DB2\NODE0000\SQL00005\sessionTS.0') BUFFERPOOL sessionBP
   DB2 connect reset
   ```

   Refer to the DB2 product documentation for details.

3. Configure the correct tablespace name and page size (DB2 row size) in the session management database configuration (Figure 9-11 on page 361).

Restart WebSphere. On startup, the session manager creates a new SESSIONS table based on the page size and tablespace name specified.

9.9.6 Single versus multi-row schemas (database persistence)

When using the single-row schema, each user session maps to a single database row. This is WebSphere's default configuration for persistent session management. With this setup, there are hard limits to the amount of user-defined, application-specific data that WebSphere Application Server can access.

When using the multi-row schema, each user session maps to multiple database rows. In a multi-row schema, each session attribute maps to a database row.

In addition to allowing larger session records, using a multi-row schema can yield performance benefits, as discussed in “Multirow persistent session management - database persistence” on page 395.

It should be stressed that switching between multi-row and single-row is not a trivial proposition.

Switching from single-row to multi-row schema

To switch from single-row to multi-row schema for sessions:

- Modify the session manager properties to switch from single to multi-row schema. You need to select the **Use Multi row schema** on the Database setting of the Session Manager window, shown in Figure 9-11 on page 361.
- Manually drop the database table or delete all the rows in the session database table.

  To drop the table:

  – Determine which data source the session manager is using. This is set in the session management distributed settings window (see 9.9.1, "Enabling database persistence" on page 359).

  – Look up the database name in the data source settings. You will need to find the JDBC provider, then the data source. The database name is in the custom settings.

  – Use the database facilities to connect to the database and drop it.
- Restart the application server or cluster.

Design considerations

Consider configuring direct single-row usage to one database and multi-row usage to another database while you verify which option suits your application's specific needs. You can do this by switching the data source used, then monitoring the performance.

Table 9-3 Single versus multi-row schemas

Programming issue	**Application scenario**
Reasons to use single-row	► You can read/write all values with just one record read/write. ► This takes up less space in a database, because you are guaranteed that each session is only one record long.
Reasons *not* to use single-row	2 MB limit of stored data per session. That is, the sum of sizes of all session attributes is limited to 2 MB.

Programming issue	Application scenario
Reasons to use multi-row	► The application can store an unlimited amount of data; that is, you are limited only by the size of the database and a 2 MB-per-record limit (so the size of each session attribute can be 2 MB). ► The application can read individual fields instead of the whole record. When large amounts of data are stored in the session but only small amounts are specifically accessed during a given servlet's processing of an HTTP request, multi-row sessions can improve performance by avoiding unneeded Java object serialization.
Reasons *not* to use multi-row	If data is small in size, you probably do not want the extra overhead of multiple row reads when everything could be stored in one row.

In the case of multi-row usage, design your application data objects so that they do not have references to each other. This is to prevent circular references.

For example, suppose you are storing two objects A and B in the session using HttpSession.put(..), and A contains a reference to B. In the multi-row case, because objects are stored in different rows of the database, when objects A and B are retrieved later, the object graph between A and B is different from that stored. A and B behave as independent objects.

9.9.7 What is written to the persistent session database

WebSphere supports two modes for writing session contents to the persistent store:

- **Only updated attributes**. Write only the HttpSession properties that have been updated via setAttribute() and removeAttribute().
- **All session attributes**. Write all the HttpSession properties to the database.

When a new session is initially created (with either of the above two options) the entire session is written, including any Java objects bound to the session. When using database persistence, the behavior for subsequent servlet or JSP requests for this session varies depending on whether the single-row or multi-row database mode is in use.

- In single-row mode:
  - Only updated attributes: If any session attribute has been updated (via setAttribute or removeAttribute), then all of the objects bound to the session will be written to the database.
  - All session attributes: All bound session attributes will be written to the database.
- In multi-row mode:
  - Only updated attributes: Only the session attributes that were specified via setAttribute or removeAttribute will be written to the database.
  - All session attributes: All of the session attributes that reside in the cache will be written to the database. If the session has never left the cache, then this should contain all of the session attributes.

By using the All session attributes mode, servlets and JSPs can change Java objects that are attributes of the HttpSession without having to call setAttribute() on the HttpSession for that Java object in order for the changes to be reflected in the database.

Adding the all session attributes mode provides some flexibility to the application programmer and protects against possible side effects of moving from local sessions to persistent sessions.

However, using All session attributes mode can potentially increase activity and be a performance drain. Individual customers will have to evaluate the pros and cons for their installation. It should be noted that the combination of All session attributes mode with time-based write could greatly reduce the performance penalty and essentially give you the best of both worlds.

As shown in Example 9-12 on page 385 and Example 9-13 on page 385, the initial session creation contains a setAttribute but subsequent requests for that session do not need to use setAttribute.

Example 9-12 Initial servlet

```
HttpSession sess = request.getSession(true);
myClass myObject = new myClass();
myObject.someInt = 1;
sess.setAttribute("myObject", myObject);   // Bind object to the session
```

Example 9-13 Subsequent servlet

```
HttpSession sess = request.getSession(false);
myObject =  sess.getAttribute("myObject");   // get bound session object
myObject.someInt++;  // change the session object
// setAttribute() not needed with write "All session attributes" specified
```

Example 9-14 and Example 9-15 show the case where setAttribute is still required even though the write all session attributes option is enabled.

Example 9-14 Initial servlet

```
HttpSession sess = request.getSession(true);
String myString = new String("Initial Binding of Session Object");
sess.setAttribute("myString", myString);   // Bind object to the session
```

Example 9-15 Subsequent servlet

```
HttpSession sess = request.getSession(false);
String myString  = sess.getAttribute("myString"); // get bound session object
...
myString = new String("A totally new String");  // get a new String object
sess.setAttribute("myString", myString);   // Need to bind the object to the
session since a NEW Object is used
```

HttpSession set/getAttribute action summary

Table 9-4 summarizes the action of the HttpSession setAttribute and removeAttribute methods for various combinations of the row type, write contents, and write frequency session persistence options.

Table 9-4 Write contents versus write frequency

Row type	Write contents	Write frequency	Action for setAttribute	Action for remove-Attribute
Single-row	Only updated attributes	End of servlet service / sync() call with Manual update	If any of the session data has changed, then write all of this session's data from cache[1]	If any of the session data has changed, then write all of this session's data from cache[1]

Row type	Write contents	Write frequency	Action for setAttribute	Action for remove-Attribute
Single-row	Only updated attributes	Time-based	If any of the session data has changed, then write all of this session's data from cache[1]	If any of the session data has changed, then write all of this session's data from cache[1]
Single-row	All session attributes	End of servlet service / sync() call with Manual update	Always write all of this session's data from cache[2]	Always write all of this session's data from cache[2]
Single-row	All session attributes	Time-based	Always write all of this session's data from cache	Always write all of this session's data from cache
Multi-row	Only updated attributes	End of servlet service / sync() call with Manual update	Write only thread-specific data that has changed	Delete only thread-specific data that has been removed
Multi-row	Only updated attributes	Time-based	Write thread-specific data that has changed for *all* threads using this session	Delete thread-specific data that has been removed for *all* threads using this session
Multi-row	All session attributes	End of servlet service / sync() call with Manual update	Write all session data from cache	Delete thread-specific data that has been removed for *all* threads using this session
Multi-row	All session attributes	Time-based	Write all session data from cache	Delete thread-specific data that has been removed for *all* threads using this session

Notes:

1. When a session is written to the database while using single-row mode, *all* of the session data is written. Therefore, no database deletes are necessary for properties that were removed via removeAttribute(), since the write of the entire session will not include removed properties.

Multi-row mode has the notion of thread-specific data. Thread-specific data is defined as session data that was added or removed while executing under this thread. If End of servlet service or Manual update modes are used and Only updated attributes is enabled, then only the thread-specific data is written to the database.

9.10 Invalidating sessions

Sessions should be invalidated when the user no longer needs the session object (for example, the user has logged off the site). Invalidating a session removes it from the session cache, as well as from the persistent store.

WebSphere offers three methods for invalidation session objects:

- Programmatically, by calling the invalidate() method on the session object. If the session object is accessed by multiple threads in a Web application, take care that none of the threads still have references to the session object.
- An invalidator thread scans for timed-out sessions every n seconds, where n is configurable from the administrative console. The session timeout setting is found in the session management configuration for the application server Web container.
- For persistent sessions, the administrator can specify times when the scan will be run. This feature has the following benefits when used with persistent session.
  - Persistent store scans can be scheduled during periods that normally have low demand. This avoids slowing down online applications due to contention in the persistent store.
  - When this setting is used with the End of servlet service write frequency option, WebSphere does not have to write out the last access time with every HTTP request. This is because WebSphere does not have to synchronize the invalidator thread's deletion with the HTTP request access.

  The session cleanup schedule setting is found in the session management settings for the Web container. You will find it with the custom tuning properties for distributed environments.

Notes:

1. HttpSession timeouts are not enforced. Instead, all invalidation processing is handled at the configured invalidation times.
2. With listeners, which are described in “Session listeners” on page 388, processing is potentially delayed by this configuration. It is not recommended if listeners are used.

9.10.1 Session listeners

Some listener classes are defined in the Servlet 2.3 specification in order to listen for state changes of a session and its attributes: for example, to create, destroy a session, and to add, delete, and modify its attributes. This allows greater control over interactions with sessions. This leads programmers to be able to monitor creation, deletion, and modification of sessions. Programmers can perform initialization tasks when a session is created or clean up tasks when a session is removed. It is also possible to perform some specific tasks for the attribute when an attribute is added/deleted or modified.

The following are the Listener interfaces to monitor the events associated with the HttpSession object:

- javax.servlet.http.HttpSessionListener

  This interface monitors creation and deletion, including session timeout, of a session.
- javax.servlet.http.HttpSessionAttributesListener

  This interface monitors changes of session attributes, such as add, delete, and replace.
- javax.servlet.http.HttpSessionActivationListener

  This interface monitors activation and passivation of sessions. This interface is useful to monitor if the session exists, whether on memory or not, when persistent session is used.

Table 9-5 on page 389 is a summary of the interfaces and methods.

Table 9-5 Listener interfaces and their methods

Target	Event	Interface	Method
session	create	HttpSessionListener	sessionCreated()
	destroy	HttpSessionListener	sessionDestroyed()
	activate	HttpSessionActivationListener	sessionDidActivate()
	passivate	HttpSessionActivationListener	sessionWillPassivate()
attribute	add	HttpSessionAttributesListener	attributeAdded()
	remove	HttpSessionAttributesListener	attributeRemoved()
	replace	HttpSessionAttributesListener	attributeReplaced()

For more information, see the API documentation at:

```
http://java.sun.com/j2ee/sdk_1.3/techdocs/api/
```

9.11 Session security

WebSphere Application Server maintains the security of individual sessions. When session manager integration with WebSphere security is enabled, the session manager checks the user ID of the HTTP request against the user ID of the session held within WebSphere. This check is done as part of the processing of the request.getSession() function. If the check fails, WebSphere throws an com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException exception. If it succeeds, the session data is returned to the calling servlet or JSP.

Session security checking works with the standard HttpSession. The identity or user name of a session can be accessed via the com.ibm.websphere.servlet.session.IBMSession interface. An unauthenticated identity is denoted by the user name "anonymous".

The session manager uses WebSphere's security infrastructure to determine the authenticated identity associated with a client HTTP request that either retrieves or creates a session. For information on WebSphere security features, see *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

Security integration rules for HTTP sessions

Session management security has the following rules:

- Sessions in unsecured pages are treated as accesses by the “anonymous” user.
- Sessions created in unsecured pages are created under the identity of that “anonymous” user.
- Sessions in secured pages are treated as accesses by the authenticated user.
- Sessions created in secured pages are created under the identity of the authenticated user. They can only be accessed in other secured pages by the same user. To protect these sessions from use by unauthorized users, they cannot be accessed from an unsecure page, that is, do not mix access to secure and unsecure pages.
- Security integration in session manager is not supported in HTTP form-based login with SWAM (Simple WebSphere Authentication Mechanism).

Table 9-6 lists possible scenarios when security integration is enabled, where outcomes depend on whether the HTTP request was authenticated and whether a valid session ID and user name was passed to the session manager.

Table 9-6 HTTP session security

Request session ID/user name	**Unauthenticated HTTP request is used to retrieve the session**	**Authenticated HTTP request is used to retrieve the session. The user ID in the HTTP request is “FRED”.**
No session ID was passed in for this request, or the ID is for a session that is no longer valid.	A new session is created. The user name is “anonymous”.	A new session is created. The user name is “FRED”.
A valid session ID is received. The current session user name is “anonymous”.	The session is returned.	The session is returned. The session manager changes the user name to “FRED”.
A valid session ID is received. The current session user name is “FRED”.	The session is not returned. UnauthorizedSession-RequestException is thrown[1].	The session is returned.

Request session ID/user name	**Unauthenticated HTTP request is used to retrieve the session**	**Authenticated HTTP request is used to retrieve the session. The user ID in the HTTP request is “FRED”.**
A valid session ID is received. The current session user name is “BOB”.	The session is not returned. UnauthorizedSession-RequestException is thrown[1].	The session is not returned. UnauthorizedSession-RequestException is thrown[1].

Note:
com.ibm.websphere.servlet.session.UnauthorizedSessionRequestException is thrown to the servlet or JSP.

See 9.7, “Advanced settings for session management” on page 349 for the steps to enable session security.

9.12 Session performance considerations

This section includes guidance for developing and administering scalable, high-performance Web applications using WebSphere Application Server session support.

9.12.1 Session size

Large session objects pose several problems for a Web application. If the site uses session caching, large sessions reduce the memory available in the WebSphere instance for other tasks, such as application execution.

For example, assume a given application stores 1 MB of information per user session object. If 100 users arrive over the course of 30 minutes, and assume the session timeout remains at 30 minutes, the application server instance must allocate 100 MB just to accommodate the newly arrived users in the session cache. Note this number does not include previously allocated sessions that have not timed out yet. The actual memory required by the session cache could be considerably higher than 100 MB.

1 MB per user session * 100 users = 100 MB

Web developers and administrators have several options for improving the performance of session management:

- Reduce the size of the session object
- Reduce the size of the session cache
- Add additional instances
- Invalidate unneeded sessions
- Increase the memory available
- Reduce the session timeout interval

Reducing session object size

Web developers must carefully consider the information kept by the session object:

- Removing information easily obtained or easily derived helps keep the session object small.
- Rigorous removal of unnecessary, unneeded, or obsolete data from the session.
- Consider whether it would be better to keep a certain piece of data in an application database rather than in the HTTP session. This gives the developer full control over when the data is fetched or stored and how it is combined with other application data. Web developers can leverage the power of SQL if the data is in an application database.

This becomes particularly important when persistent sessions are used. There is a significant performance overhead when WebSphere has to serialize a large amount of data and write it to the persistent store. Even if the Write contents option is enabled, if the session object contains large Java objects or collections of objects that are updated regularly, there is a significant performance penalty in persisting these objects. This penalty can be reduced by using time-based writes.

Notes: In general the best performance will be realized with session objects that are less than 2 KB in size. Once the session object starts to exceed 4-5 KB in size, a significant decrease in performance can be expected.

Even if session persistence is not an issue, minimizing the session object size will help to protect your Web application from scale-up disasters as user numbers increase. Large session objects will require more and more JVM memory, leaving no room to run servlets.

See 9.9.5, “Using larger DB2 page sizes (database persistence)” on page 380 to find out how WebSphere can provide faster persistence of larger session objects when using DB2.

Session cache size

The session manager allows administrators to change the session cache size to alter the cache's memory footprint. By default, the session cache holds 1000 session objects. By lowering the number of session objects in the cache, the administrator reduces the memory required by the cache.

However, if the user's session is not in the cache, WebSphere must retrieve it from either the overflow cache (for local caching) or the session database (for persistent sessions). If the session manager must retrieve persistent sessions frequently, the retrievals may impact overall application performance.

WebSphere maintains overflowed local sessions in memory, as discussed in 9.6, "Local sessions" on page 348. Local session management with cache overflow enabled allows an unlimited number of sessions in memory. In order to limit the cache footprint to the number of entries specified in session manager, the administrator should use persistent session management, or disable overflow.

Note: When using local session management without specifying the Allow overflow property, a full cache will result in the loss of user session objects.

Creating additional application server instances

WebSphere also gives the administrator the option of creating additional application server instances. Creating additional instances spread the demand for memory across more JVMs, thus reducing the memory burden on any particular instance. Depending on the memory and CPU capacity of the machines involved, the administrator may add additional instances within the same machine. Alternatively, the administrator may add additional machines to form a hardware cluster, and spread the instances across this cluster.

Note: When configuring a session cluster, session affinity routing provides the most efficient strategy for user distribution within the cluster, even with session persistence enabled. With cluster members, the Web server plug-in provides affinity routing among cluster member instances.

Invalidating unneeded sessions

If the user no longer needs the session object, for example, when the user has logged out of the site, it should be invalidated. Invalidating a session removes it from the session cache, as well as from the session database. For more information see 9.10, "Invalidating sessions" on page 387.

Increasing available memory

WebSphere allows the administrator to increase an application server's heap size. By default, WebSphere allocates 256 MB as the maximum heap size. Increasing this value allows the instance to obtain more memory from the system, and thus hold a larger session cache.

A practical limit exists, however, for an instance's heap size. The machine memory containing the instance needs to support the heap size requested. Also, if the heap size grows too large, the length of the garbage collection cycle with the JVM may impact overall application performance. This impact has been reduced with the introduction of multi-threaded garbage collection.

Session timeout interval

By default, each user receives a 30-minute interval between requests before the session manager invalidates the user's session. Not every site requires a session timeout interval this generous. By reducing this interval to match the requirements of the average site user, the session manager purges the session from the cache (and the persistent store, if enabled) more quickly.

Avoid setting this parameter too low and frustrating users. The administrator must take into account a reasonable time for an average user to interact with the site (read returned data, fill out forms, and so on) when setting the interval. Also, the interval must represent any increased response time during peak times on the site (such as heavy trading days on a brokerage site, for example).

Finally, in some cases where the persistent store contains a large number of entries, frequent execution of the timeout scanner reduces overall performance. In cases where the persistent store contains many session entries, avoid setting the session timeout so low it triggers frequent, expensive scans of the persistent store for timed-out sessions. Alternatively, the administrator should consider schedule-based invalidation where scans for invalid object can be deferred to a time that normally has low demand. See 9.10, "Invalidating sessions" on page 387.

9.12.2 Reducing persistent store I/O

From a performance point of view, the Web developer should consider the following:

- Optimize the use of the HttpSession within a servlet. Only store the minimum amount of data required in HttpSession. Data that does not have to be recovered after a cluster member fails or is shut down may be best kept elsewhere, such as in a hash table. Recall that HttpSession is intended to be used as a *temporary* store for state information between browser invocations.

- JSPs that do not need to access the session object should specify "session=false" in the JSP directive.
- Use time-based write frequency mode. This greatly reduces the amount of I/O, since the persistent store updates are deferred up to a configurable number of seconds. Using this mode, all of the outstanding updates for a Web application are written out periodically based on the configured write interval.
- Use the **Specify distributed sessions cleanup schedule** option. When using the End of servlet service write frequency mode, WebSphere does not have to write out the last access time with every HTTP request. This is because WebSphere does not have to synchronize the invalidator thread's deletion with the HTTP request's access.

9.12.3 Multirow persistent session management - database persistence

When a session contains multiple objects accessed by different servlets or JSPs in the same Web application, multi-row session support provides a mechanism for improving performance. Multi-row session support stores session data in the persistent session database by Web application and value. Table 9-7 shows a simplified representation of a multi-row database table.

Table 9-7 Simplified multi-row session representation

Session ID	Web application	Property	Small value	Large value
DA32242SSGE2	ShoeStore	ShoeStore.First.Name	"Alice"	
DA32242SSGE2	ShoeStore	ShoeStore.Last.Name	"Smith"	
DA32242SSGE2	ShoeStore	ShoeStore.Big.String		"A big string...."

In this example, if the user visits the ShoeStore application, and the servlet involved needs the user's first name, the servlet retrieves this information through the session API. The session manager brings into the session cache only the value requested. The ShoeStore.Big.String item remains in the persistent session database until the servlet requests it. This saves time by reducing both the data retrieved and the serialization overhead for data the application does not use.

After the session manager retrieves the items from the persistent session database, these items remain in the in-memory session cache. The cache accumulates the values from the persistent session database over time as the various servlets within the Web application request them. With WebSphere's session affinity routing, the user returns to this same cached session instance repeatedly. This reduces the number of reads against the persistent session database, and gives the Web application better performance.

How session data is written to the persistent session database has been made configurable in WebSphere. For information on session updates using single and multi-row session support, see 9.9.6, "Single versus multi-row schemas (database persistence)" on page 381. Also see 9.9.7, "What is written to the persistent session database" on page 383.

Even with multi-row session support, Web applications perform best if the overall contents of the session objects remain small. Large values in session objects require more time to retrieve from the persistent session database, generate more network traffic in transit, and occupy more space in the session cache after retrieval.

Multi-row session support provides a good compromise for Web applications requiring larger sessions. However, single-row persistent session management remains the best choice for Web applications with small session objects. Single-row persistent session management requires less storage in the database, and requires fewer database interactions to retrieve a session's contents (all of the values in the session are written or read in one operation). This keeps the session object's memory footprint small, as well as reducing the network traffic between WebSphere and the persistent session database.

Note: Avoid circular references within sessions if using multi-row session support. The multi-row session support does not preserve circular references in retrieved sessions.

9.12.4 Managing your session database connection pool

When using persistent session management, the session manager interacts with the defined database through a WebSphere Application Server data source. Each data source controls a set of database connections known as a connection pool. By default, the data source opens a pool of no more than 10 connections. The maximum pool size represents the number of simultaneous accesses to the persistent session database available to the session manager.

For high-volume Web sites, the default settings for the persistent session data source may not be sufficient. If the number of concurrent session database accesses exceeds the connection pool size, the data source queues the excess requests until a connection becomes available. Data source queueing can impact the overall performance of the Web application (sometimes dramatically).

For best performance, the overhead of the connection pool used by the session manager needs to be balanced against the time that a client may spend waiting for an in-use connection to become available for use. By definition a connection pool is a *shared* resource, so in general the best performance will typically be realized with a connection pool that has significantly fewer connections than the number of simultaneous users.

A large connection pool does not necessarily improve application performance. Each connection represents memory overhead. A large pool decreases the memory available for WebSphere to execute applications. Also, if database connections are limited because of database licensing issues, the administrator must share a limited number of connections among other Web applications requiring database access as well. This is one area where performance tuning tests will be required to determine the optimal setting for a given application.

As discussed above, session affinity routing combined with session caching reduces database read activity for session persistence. Likewise, manual update write frequency, time-based write frequency, and multi-row persistent session management reduce unnecessary writes to the persistent database. Incorporating these techniques may also reduce the size of the connection pool required to support session persistence for a given Web application.

Prepared statement caching is a connection pooling mechanism that can be used to further improve session database response times. A cache of previously prepared statements is available on a connection. When a new prepared statement is requested on a connection, the cached prepared statement is returned, if available. This caching reduces the number of costly prepared statements created, which improves response times.

In general, base the prepared statement cache size on:

- The smaller of:
  - Number of concurrent users
  - Connection pool size
- The number of different prepared statements

With 50 concurrent users, a connection pool size of 10, and each user using 2 statements (a select and an insert) the prepared statement cache size should be at least 10 x 2 = 20 statements. To find out more, see the “Prepared statement cache size” article in the *WebSphere Tuning Guide* (included with the InfoCenter).

9.12.5 Session database tuning

While the session manager implementation in WebSphere provides for a number of parameters that can be tuned to improve performance of applications that utilize HTTP sessions, maximizing performance will require tuning the underlying session persistence table. WebSphere provides a “first step” in this regard by creating an index for the sessions table when creating the table. The index is comprised of the session ID, the property ID (for multi-row sessions), and the Web application name.

While most database managers provide a great deal of capability in tuning at the table or tablespace level, creation of a separate database or instance will afford the most flexibility in tuning. Proper tuning of the instance/database can improve performance by 5% (or more) over that which can be achieved by simply tuning the table or tablespace.

While the specifics will vary depending on the database and operating system in use, in general the database should be tuned and configured as appropriate for a database that experiences a great deal of I/O. The DBA should monitor and tune the database buffer pools, database log size, and write frequency. Additionally, maximizing performance will require striping the database/instance across multiple disk drives and disk controllers, and utilizing any hardware or OS buffering that is available in order to reduce disk contention.

10

WebSphere naming implementation

In this chapter, we describe the concepts behind the naming functionality provided as part of IBM WebSphere Application Server. We cover:

- New features in WebSphere Application Server V5
- WebSphere naming architecture
- Interoperable Naming Service (INS)
- Distributed CosNaming
- Configured bindings
- Initial contexts
- Federation of name spaces
- Interoperability
- Examples
- Naming tools
- Configuration

10.1 New features

The following are new features of a WebSphere Application Server V5 name space:

- The name space is distributed

  For additional scalability, the name space for a cell is distributed among the various servers. The Deployment Manager, node agent and application server processes all host a name server. In previous releases, there was only a single name server for an entire administrative domain.

  In WebSphere releases prior to V5, all servers shared the same default initial context, and everything was bound relative to that initial context. In WebSphere Application Server V5, the default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server they are associated with.

- Transient and persistent partitions

  The name space is partitioned into transient areas and persistent areas. Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell persistent root, which can be used for cell-scoped persistent bindings, and a node persistent root, which can be used to bind objects with a node scope.

- Federated name space structure

  A name space is a collection of all names bound to a particular name server. A name space can contain naming context bindings to contexts located in other servers. If this is the case, the name space is said to be a *federated name space*, because it is a collection of name spaces from multiple servers. The name spaces link together to cooperatively form a single logical name space.

  In a federated name space, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

  The name space for the WebSphere Application Server V5 cell is federated among the Deployment Manager, node agents, and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell name space, with the various server roots and persistent partitions of the name space being interconnected by means of the single logical name space.

- Configured bindings

  The configuration graphical interface and script interfaces can be used to configure bindings in various root contexts within the name space. These bindings are read-only and are bound by the system at server startup.

- Support for CORBA Interoperable Naming Service (INS) object URLs

  WebSphere Application Server V5 contains support for CORBA object URLs (corbaloc and corbaname) as JNDI provider URLs and lookup names, as required by the J2EE 1.3 specification.

10.2 WebSphere naming architecture

The WebSphere naming implementation is composed of JNDI and CosNaming. JNDI provides the client-side access to naming and presents the programming model used by application developers. CosNaming provides the server-side implementation and is where the name space is actually stored. JNDI essentially provides a client-side wrapper of the name space stored in CosNaming, and interacts with the CosNaming server on behalf of the client.

The model of JNDI over CosNaming has existed in several releases of WebSphere. With J2EE 1.3, limited CosNaming functionality is now required for interoperability between application servers from different vendors. This level of CosNaming, known as Interoperable Naming Service (INS), has been added for WebSphere Application Server V5.

The following subsections provide a summary of the WebSphere naming architecture, its federated name space, and its support for JNDI.

For an explanation of the WebSphere implementations of INS and Distributed CosNaming, see 10.3, “Interoperable Naming Service (INS)” on page 414 and 10.4, “Distributed CosNaming” on page 417 respectively.

10.2.1 Components

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications, such as EJB homes. These objects are bound into a mostly hierarchical structure, referred to as a *name space*. In this structure, all non-leaf objects are called *contexts*. Leaf objects can be contexts and other types of objects.

The name space structure consists of a set of *name bindings*, each consisting of a name relative to a specific context and the object bound with that name. For example, the name “myApp/myEJB” consists of one non-leaf binding with the name “myApp,” which is a context. The name also includes one leaf binding with the name “myEJB,” relative to “myApp.” The object bound with the name “myEJB” in this example happens to be an EJB home reference. The whole name “myApp/myEJB” is relative to the initial context, which can be viewed as a starting place when performing naming operations.

The name space can be accessed and manipulated through a *name server*. Users of a name server are referred to as naming clients. Naming clients typically use Java Naming and Directory Interface (JNDI) to perform naming operations. Naming clients can also use the Common Object Request Broker Architecture (CORBA) CosNaming interface.

Figure 10-1 summarizes the naming architecture and its components.

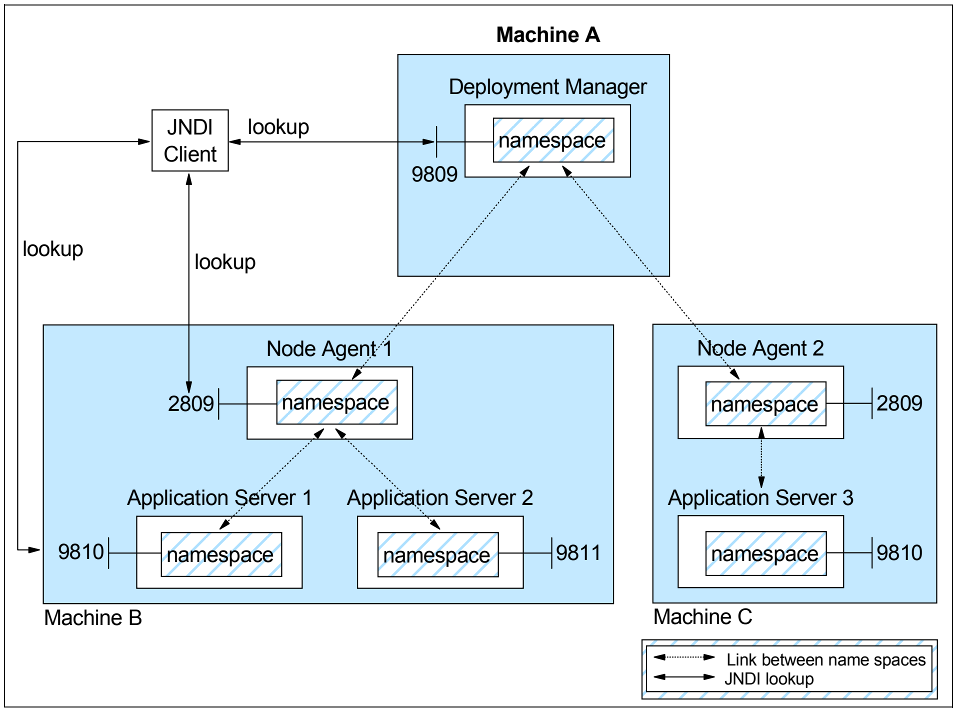


Figure 10-1 Naming topology

Notice that all WebSphere Application Server V5 processes (except the JMS server) host their own Naming Service and local name space.

10.2.2 JNDI support

Each IBM WebSphere Application Server managed process (JVM) includes:

- A name server to provide shared access to its components.
- An implementation of the javax.naming JNDI package, which allows users to access the WebSphere name server through the JNDI naming interface.

Note: The JNDI implementation provided by IBM WebSphere Application Server is based on Version 1.2 of the JNDI interface, and was tested with Version 1.2.1 of Sun's JNDI SPI (Service Provider Interface).

IBM WebSphere Application Server does not provide implementations for the following Java extension packages:

- javax.naming.directory
- javax.naming.ldap

Nor does it support interfaces defined in the javax.naming.event package.

However, to provide access to LDAP servers, the JDK shipped with IBM WebSphere Application Server supports Sun's implementation of:

- javax.naming.ldap
- com.sun.jndi.ldap.LdapCtxFactory

10.2.3 JNDI bindings

There are three options available for binding EJB (<ejb-ref>) and resource (<resource-ref>) object names to the IBM WebSphere Application Server name space:

- Simple name
- Corbaname
- Compound/fully qualified name

Simple name

The WebSphere Application Server V4 style of simple name binding is guaranteed to succeed only on a single server. It can be used in a servlet or EJB, if it is certain that the object being looked up is located on the same application server.

Example 10-1 Simple name form of JNDI binding

```
ejb/webbank/Account
```

Corbaname

The corbaname binding is always guaranteed to work. However, it requires that the correct path to the object be known at deployment time.

Example 10-2 Corbaname form of JNDI binding

```
corbaname::myhost1:9812/NameServiceServerRoot#ejb/webbank/Account
```

Compound name

The fully qualified (compound name) JNDI name is always guaranteed to work.

Example 10-3 Compound name form of JNDI binding

```
cell/nodes/node1/servers/server1/ejb/webbank/Account
```

Note: Using compound names for JNDI bindings is recommended for WebSphere Application Server V5.

10.2.4 Federated name space

Figure 10-2 provides an overview of the federated name space used in IBM WebSphere Application Server.

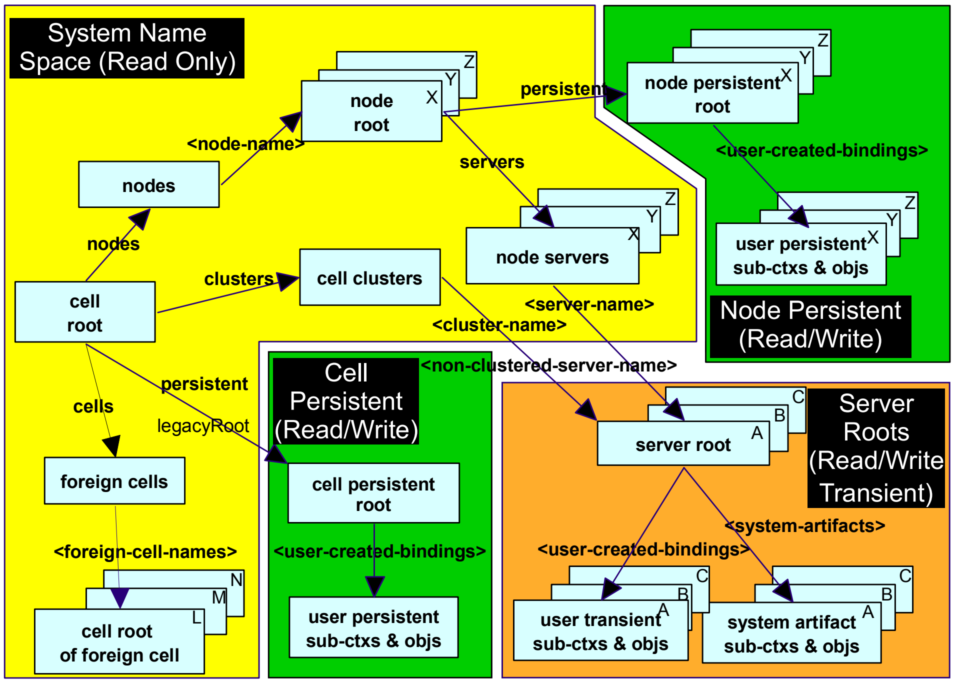


Figure 10-2 Federated name space

The name space can be broken down into a number distinct partitions that differ in whether they are updateable and/or persistent.

System partition

The system partition is a reflection of the topology and is read-only. That is, this part of the name space cannot be changed programmatically, because it is based on the configuration rather than runtime settings.

This partition of the name space is persistent, with the data stored in the XML repository containing the topological information.

Node and cell persistent partitions

The persistent partitions are meant primarily for the storage of resource configuration, such as data sources, JMS destinations etc. This data can be modified using accessing the JNDI APIs directly, or via the administration clients (which access the APIs on the user's behalf). The persistent data is stored to a group of XML files.

There are two persistent partitions in the federated name space:

- Cell persistent root

  Used to register persistent objects that are available to all the nodes and managed processes of a cell. A binding created under the cell persistent root is saved as part of the cell configuration and continues to exist until it is explicitly removed.

  Applications that need to create additional persistent object bindings associated with the cell can bind those objects under the cell persistent root.

  Note: The cell persistent root is not designed for transient, rapidly changing bindings. Instead, the bindings should be more static in nature, such as part of application setup or configuration, and not created at runtime.

  In order to bind objects to the cell persistent root, the Deployment Manager and all node agents in the cell must be running.

  An important role of the cell persistent root is as the initial context for clients running in previous WebSphere releases. If an EJB is to be accessed by WebSphere Application Server V4.x or V3.5.x clients, a binding for it must be added to the cell persistent root. See 10.5, "Configured bindings" on page 418 for details.

- Node persistent root

  Used to register persistent objects that are available to the nodes and its managed processes. Similar to the cell partition except that each node has its own node persistent root. A binding created under a node persistent root is saved as part of that node's configuration and continues to exist until it is explicitly removed.

Applications that need to create additional persistent object bindings associated with a specific node can bind those objects under that particular node's node persistent root.

Note: The node persistent area is not designed for transient, rapidly changing bindings. Instead, the bindings should be more static in nature, such as part of application setup or configuration, and not created at runtime.

In order to bind objects to a node persistent root, the node agent for the node must be running.

In the federated name space, there is no node root for the Deployment Manager node since no node agent or application servers run in that node.

The node persistent root plays no special role in interoperability with WebSphere clients of previous releases.

Transient partitions

The server root transient partition in Figure 10-2 on page 404 is updateable through APIs, and is meant for information such as EJB bindings and JNDI names. This name space is transient. Each time a server process starts, it reads settings from the file system, for example EJB deployment descriptors, to register the necessary objects in this space.

Note: The Naming Service of each managed process is a listener to configuration changes. This means that the local name space will be updated automatically when configuration changes occur. For example, there is no need to restart a node agent to update its name space when a new application server is created.

10.2.5 Local name space structure

The structure of the federated name space is hierarchical, with each process' local name space federated/linked using corbaloc URLs that allow transparent name searches both within a single local name space and from one local name space to another.

The contents of the cell, node, and process local name spaces are described in the following sections.

Cell-level name space

The cell-level name space, hosted by the Deployment Manager, has a structure as shown in Example 10-4.

Note: (top) represents the root of the federated name space.

Example 10-4 Cell name space dump (dumpNameSpace -port <dmgr_bootstrap>)

```
   1 (top)
   2 (top)/persistent
   3 (top)/persistent/cell
         Linked to context: NodeANetwork
   4 (top)/legacyRoot
         Linked to context: NodeANetwork/persistent
   5 (top)/domain
         Linked to context: NodeANetwork
   6 (top)/cells

   7 (top)/clusters
   8 (top)/clusters/MyCluster
Linked to URL:corbaloc::NodeA:9812,:NodeA:9811/NameServiceServerRoot

   9 (top)/cellname
  10 (top)/cell
         Linked to context: NodeANetwork

  11 (top)/deploymentManager
     Linked to context: NodeANetwork/nodes/NodeAManager/servers/dmgr

  12 (top)/nodes
  13 (top)/nodes/NodeAManager
  14 (top)/nodes/NodeAManager/nodename
  15 (top)/nodes/NodeAManager/domain
         Linked to context: NodeANetwork
  16 (top)/nodes/NodeAManager/cell
  16    Linked to context: NodeANetwork
  17 (top)/nodes/NodeAManager/servers
  18 (top)/nodes/NodeAManager/servers/dmgr
  19 (top)/nodes/NodeAManager/servers/dmgr/distributedmap
          com.ibm.websphere.cache.DistributedMap
  20 (top)/nodes/NodeAManager/servers/dmgr/thisNode
         Linked to context: NodeANetwork/nodes/NodeAManager
  21 (top)/nodes/NodeAManager/servers/dmgr/cell
         Linked to context: NodeANetwork
  22 (top)/nodes/NodeAManager/servers/dmgr/jta
  23 (top)/nodes/NodeAManager/servers/dmgr/jta/usertransaction
  24 (top)/nodes/NodeAManager/servers/dmgr/jdbc
  25 (top)/nodes/NodeAManager/servers/dmgr/jdbc/Sessions
```

```
       sessdb
26 (top)/nodes/NodeAManager/servers/dmgr/TransactionFactory
       com.ibm.ejs.jts.jts.ControlSet$LocalFactory
27 (top)/nodes/NodeAManager/servers/dmgr/servername
28 (top)/nodes/NodeAManager/node
       Linked to context: NodeANetwork/nodes/NodeAManager
29 (top)/nodes/NodeB
30 (top)/nodes/NodeB/cell
       Linked to context: NodeANetwork
31 (top)/nodes/NodeB/servers
32 (top)/nodes/NodeB/servers/jmsserver
32    Linked to URL: corbaloc::NodeB:2810/NameServiceServerRoot
33 (top)/nodes/NodeB/servers/nodeagent
       Linked to URL: corbaloc::NodeB:2809/NameServiceServerRoot
34 (top)/nodes/NodeB/servers/server1
       Linked to URL: corbaloc::NodeB:9810/NameServiceServerRoot
35 (top)/nodes/NodeB/node
       Linked to context: NodeANetwork/nodes/NodeB
36 (top)/nodes/NodeB/nodeAgent
       Linked to URL: corbaloc::NodeB:2809/NameServiceServerRoot
37 (top)/nodes/NodeB/persistent
      Linked to URL: corbaname::NodeB:2809/NameServiceNodeRoot#persistent
38 (top)/nodes/NodeB/nodename
39 (top)/nodes/NodeB/domain
       Linked to context: NodeANetwork
40 (top)/nodes/NodeA
41 (top)/nodes/NodeA/persistent
     Linked to URL: corbaname::NodeA:2809/NameServiceNodeRoot#persistent
42 (top)/nodes/NodeA/nodename
43 (top)/nodes/NodeA/domain
       Linked to context: NodeANetwork
44 (top)/nodes/NodeA/cell
        Linked to context: NodeANetwork
45 (top)/nodes/NodeA/servers
46 (top)/nodes/NodeA/servers/jmsserver
       Linked to URL: corbaloc::NodeA:2810/NameServiceServerRoot
47 (top)/nodes/NodeA/servers/nodeagent
       Linked to URL: corbaloc::NodeA:2809/NameServiceServerRoot
48 (top)/nodes/NodeA/servers/MyClusterServer2
       Linked to URL: corbaloc::NodeA:9812/NameServiceServerRoot
49 (top)/nodes/NodeA/servers/MyClusterServer1
       Linked to URL: corbaloc::NodeA:9811/NameServiceServerRoot
50 (top)/nodes/NodeA/servers/server1
       Linked to URL: corbaloc::NodeA:9810/NameServiceServerRoot
51 (top)/nodes/NodeA/node
       Linked to context: NodeANetwork/nodes/NodeA
52 (top)/nodes/NodeA/nodeAgent
       Linked to URL: corbaloc::NodeA:2809/NameServiceServerRoot
```

The cell-level name space contains:

- A link to the cell persistent root, /persistent/cell.
- A hierarchy of contexts for nodes and the servers managed by each node.
- A full set of entries for the Deployment Manager node (NodeAManager) and the Deployment Manager server (dmgr).
- The objects registered in JNDI by the dmgr server.
- A corbaloc URL link to the local name space of each of the other nodes in the cell.
- A number of cross links for the federated name space:
  - /cells
  - /clusters
  - /legacyRoot

Note: Because of the hierarchical structure of the cell/node/server relationship, the following naming conventions and constraints exist:

1. No two nodes can have the same name. Node names must be unique within a cell.
2. Two application servers on different nodes can have the same name.
3. Two application servers on the same node must have different names. Application server names must be unique within a node.
4. The JMS server is called *jmsserver* on each node.

Node-level name space

A node-level name space, hosted by a node agent, has a structure as shown in Example 10-5.

Example 10-5 Node-level name space (dumpNameSpace -port <NodeA_bootstrap>)

```
    1 (top)
    2 (top)/persistent
    3 (top)/persistent/cell
         Linked to context: NodeANetwork
    4 (top)/legacyRoot
         Linked to context: NodeANetwork/persistent
    5 (top)/domain
         Linked to context: NodeANetwork
    6 (top)/cells
    7 (top)/clusters
    8 (top)/clusters/MyCluster
         Linked to URL: corbaloc::NodeA:9812,:NodeA:9811/NameServiceServerRoot
    9 (top)/cellname
```

```
10 (top)/cell
        Linked to context: NodeANetwork
11 (top)/deploymentManager
       Linked to URL: corbaloc::NodeA:9917/NameServiceServerRoot
12 (top)/nodes
13 (top)/nodes/NodeAManager
14 (top)/nodes/NodeAManager/nodename
15 (top)/nodes/NodeAManager/domain
       Linked to context: NodeANetwork
16 (top)/nodes/NodeAManager/cell
       Linked to context: NodeANetwork
17 (top)/nodes/NodeAManager/servers
18 (top)/nodes/NodeAManager/servers/dmgr
       Linked to URL: corbaloc::NodeA:9917/NameServiceServerRoot
19 (top)/nodes/NodeAManager/node
       Linked to context: NodeANetwork/nodes/NodeAManager
20 (top)/nodes/NodeB
21 (top)/nodes/NodeB/cell
       Linked to context: NodeANetwork
22 (top)/nodes/NodeB/servers
23 (top)/nodes/NodeB/servers/jmsserver
       Linked to URL: corbaloc::NodeB:2810/NameServiceServerRoot
24 (top)/nodes/NodeB/servers/nodeagent
       Linked to URL: corbaloc::NodeB:2809/NameServiceServerRoot
25 (top)/nodes/NodeB/servers/server1
       Linked to URL: corbaloc::NodeB:9810/NameServiceServerRoot
26 (top)/nodes/NodeB/node
       Linked to context: NodeANetwork/nodes/NodeB
27 (top)/nodes/NodeB/nodeAgent
       Linked to URL: corbaloc::NodeB:2809/NameServiceServerRoot
28 (top)/nodes/NodeB/persistent
       Linked to URL: corbaname::NodeB:2809/NameServiceNodeRoot#persistent
29 (top)/nodes/NodeB/nodename
30 (top)/nodes/NodeB/domain
       Linked to context: NodeANetwork
31 (top)/nodes/NodeA
32 (top)/nodes/NodeA/persistent
33 (top)/nodes/NodeA/nodename
34 (top)/nodes/NodeA/domain
       Linked to context: NodeANetwork
35 (top)/nodes/NodeA/cell
       Linked to context: NodeANetwork
36 (top)/nodes/NodeA/servers
37 (top)/nodes/NodeA/servers/jmsserver
       Linked to URL: corbaloc::NodeA:2810/NameServiceServerRoot
38 (top)/nodes/NodeA/servers/nodeagent
39 (top)/nodes/NodeA/servers/nodeagent/servername
40 (top)/nodes/NodeA/servers/nodeagent/thisNode
       Linked to context: NodeANetwork/nodes/NodeA
```

```
41 (top)/nodes/NodeA/servers/nodeagent/cell
       Linked to context: NodeANetwork
42 (top)/nodes/NodeA/servers/MyClusterServer2
      Linked to URL: corbaloc::NodeA:9812/NameServiceServerRoot
43 (top)/nodes/NodeA/servers/MyClusterServer1
      Linked to URL: corbaloc::NodeA:9811/NameServiceServerRoot
44 (top)/nodes/NodeA/servers/server1
      Linked to URL: corbaloc::NodeA:9810/NameServiceServerRoot
45 (top)/nodes/NodeA/node
     Linked to context: NodeANetwork/nodes/NodeA
46 (top)/nodes/NodeA/nodeAgent
      Linked to context: NodeANetwork/nodes/NodeA/servers/nodeagent
```

The node-level name space contains:

- A full set of entries for the node and the node agent server (<nodename> or nodeAgent).
- A corbaloc URL link to the local name space root (NameServiceServerRoot) of each application server managed by the node.

  Note: JMS servers are not registered in the federated name space, since they do not host a local name space.

- Links to other nodes in the cell and to the servers on those nodes.
- A corbaloc URL link to the root of the Deployment Manager local name space (NameServiceServerRoot).
- A number of cross-links for the federated name space:
  - /cells
  - /clusters
  - /legacyRoot
- A link to the cell persistent root, /persistent/cell.
- A link to the node persistent root, /nodes/<nodename>/persistent.

Managed process-level name space

A process-level name space, hosted by a managed process, has a structure as shown in Example 10-6.

Example 10-6 Managed process-level name space (dumpNameSpace -port <srvr_bootstrap>)

```
 1 (top)
 2 (top)/persistent
 3 (top)/persistent/cell
```

```
  3    Linked to context:  NodeANetwork
  4 (top)/legacyRoot
  4    Linked to context:  NodeANetwork/persistent
  5 (top)/domain
  5    Linked to context:  NodeANetwork
  6 (top)/cells
  7 (top)/clusters
  8 (top)/clusters/MyCluster
Linked to URL: corbaloc:: NodeA:9812,: NodeA:9811/NameServiceServerRoot
  9 (top)/cellname
 10 (top)/cell
        Linked to context:  NodeANetwork
 11 (top)/deploymentManager
        Linked to URL: corbaloc:: NodeA:9917/NameServiceServerRoot
 12 (top)/nodes
 13 (top)/nodes/ NodeAManager
 14 (top)/nodes/ NodeAManager/nodename
 15 (top)/nodes/ NodeAManager/domain
        Linked to context:  NodeANetwork
 16 (top)/nodes/ NodeAManager/cell
        Linked to context:  NodeANetwork
 17 (top)/nodes/ NodeAManager/servers
 18 (top)/nodes/ NodeAManager/servers/dmgr
       Linked to URL: corbaloc:: NodeA:9917/NameServiceServerRoot
 19 (top)/nodes/ NodeAManager/node
       Linked to context:  NodeANetwork/nodes/ NodeAManager
 20 (top)/nodes/ NodeB
 21 (top)/nodes/ NodeB/cell
       Linked to context:  NodeANetwork
 22 (top)/nodes/ NodeB/servers
 23 (top)/nodes/ NodeB/servers/jmsserver
       Linked to URL: corbaloc:: NodeB:2810/NameServiceServerRoot
 24 (top)/nodes/ NodeB/servers/nodeagent
       Linked to URL: corbaloc:: NodeB:2809/NameServiceServerRoot
 25 (top)/nodes/ NodeB/servers/server1
 26 (top)/nodes/ NodeB/servers/server1/TransactionFactory
        com.ibm.ejs.jts.jts.ControlSet$LocalFactory
 27 (top)/nodes/ NodeB/servers/server1/distributedmap
        com.ibm.websphere.cache.DistributedMap
 28 (top)/nodes/ NodeB/servers/server1/servername
 29 (top)/nodes/ NodeB/servers/server1/thisNode
        Linked to context:  NodeANetwork/nodes/ NodeB
 30 (top)/nodes/ NodeB/servers/server1/cell
        Linked to context:  NodeANetwork
 31 (top)/nodes/ NodeB/servers/server1/webbank
 32 (top)/nodes/ NodeB/servers/server1/webbank/ejb
 33 (top)/nodes/ NodeB/servers/server1/webbankds     webbankds
 34 (top)/nodes/ NodeB/servers/server1/eis
 35 (top)/nodes/ NodeB/servers/server1/eis/DefaultDatasource_CMP
```

```
       Default_CF
36 (top)/nodes/ NodeB/servers/server1/eis/webbankds_CMP
       webbankds_CF
37 (top)/nodes/ NodeB/servers/server1/jta
38 (top)/nodes/ NodeB/servers/server1/jta/usertransaction
39 (top)/nodes/ NodeB/servers/server1/DefaultDatasource
       Default Datasource
40 (top)/nodes/ NodeB/servers/server1/jdbc
41 (top)/nodes/ NodeB/servers/server1/jdbc/Sessions sessdb
42 (top)/nodes/ NodeB/node
       Linked to context:  NodeANetwork/nodes/ NodeB
43 (top)/nodes/ NodeB/nodeAgent
       Linked to URL: corbaloc:: NodeB:2809/NameServiceServerRoot
44 (top)/nodes/ NodeB/persistent
45 (top)/nodes/ NodeB/nodename
46 (top)/nodes/ NodeB/domain
       Linked to context:  NodeANetwork
47 (top)/nodes/ NodeA
48 (top)/nodes/ NodeA/persistent
     Linked to URL: corbaname:: NodeA:2809/NameServiceNodeRoot#persistent
49 (top)/nodes/ NodeA/nodename
50 (top)/nodes/ NodeA/domain
       Linked to context:  NodeANetwork
51 (top)/nodes/ NodeA/cell
       Linked to context:  NodeANetwork
52 (top)/nodes/ NodeA/servers
53 (top)/nodes/ NodeA/servers/jmsserver
       Linked to URL: corbaloc:: NodeA:2810/NameServiceServerRoot
54 (top)/nodes/ NodeA/servers/nodeagent
       Linked to URL: corbaloc:: NodeA:2809/NameServiceServerRoot
55 (top)/nodes/ NodeA/servers/MyClusterServer2
       Linked to URL: corbaloc:: NodeA:9812/NameServiceServerRoot
56 (top)/nodes/ NodeA/servers/MyClusterServer1
       Linked to URL: corbaloc:: NodeA:9811/NameServiceServerRoot
57 (top)/nodes/ NodeA/servers/server1
       Linked to URL: corbaloc:: NodeA:9810/NameServiceServerRoot
58 (top)/nodes/ NodeA/node
       Linked to context:  NodeANetwork/nodes/ NodeA
59 (top)/nodes/ NodeA/nodeAgent
       Linked to URL: corbaloc:: NodeA:2809/NameServiceServerRoot
```

The process-level name space contains:

- A full set of entries for objects registered in the local name space of the process. These entries include resources (JDBC, JMS, etc.) read from the resources.xml of the process, as well as those registered at runtime by applications, for example EJB homes.

- A corbaloc URL link to the local name space root (NameServiceServerRoot) of the node agent.
- A corbaloc URL link to the root of the Deployment Manager local name space (NameServiceServerRoot).
- A number of cross-links for the federated name space:
  - /cells
  - /clusters
  - /legacyRoot
- A link to the cell persistent root, /persistent/cell.
- A link to the node persistent root, /nodes/<nodename>/persistent.

10.3 Interoperable Naming Service (INS)

In order to meet the requirements of the J2EE 1.3 specification, support for Interoperable Naming Service (INS) CosNaming is required. The INS allows J2EE application servers to deal with and understand names formulated according to the CORBA 2.3 naming scheme. The main advantage of INS is that it improves interoperability with other application server products, as well as CORBA servers.

The naming architecture of WebSphere Application Server V5 is compliant with the Interoperable Naming Service (INS).

The requirements of INS CosNaming include:

- *corbaloc* and *corbaname* URLs must be supported, in addition to the IIOP URL supported in WebSphere Application Server V4.
  - corbaloc

    Designates an endpoint, such as a host machine.
  - corbaname

    Designates an object's name.
- The default bootstrap port must be 2809, as compared to the default of 900 used in earlier versions of IBM WebSphere Application Server.

10.3.1 Bootstrap ports

Every WebSphere Application Server V5 process, except the JMS server, has a bootstrap server and hence port assignment.

Each process on a given machine (and WebSphere logical node) requires unique ports, including the bootstrap port. The default port assignments are:

- Application server

  Default for application server is 9810. Each subsequently created application server will be assigned a unique port number that does not conflict.

- Network deployment

  Default for node agent is 2809. Application servers are each assigned a unique non-default port, either explicitly by the administrator, or automatically determined by the administration tool.

10.3.2 CORBA URLs

CORBA URL syntax (both corbaloc and corbaname) is supported by IBM WebSphere Application Server.

corbaloc

The corbaloc form of the CORBA 2.3 URL has the following syntax:

```
corbaloc:<protocol>:<addresslist>/<key>
```

Table 10-1 corbaloc options

Setting	Description
`protocol`	The protocol used for the communication. Currently, the only valid value is iiop.
`addresslist`	List of one or more addresses (host name and port number). The addresses are separated by commas, and each address has a colon prefix.
`key`	Type of root to access. See Table 10-5 on page 424 for details.

The following examples illustrate how the corbaloc URL can range from simple to complex, depending upon whether fault tolerance (request retry with second, third etc. server) is required.

Example 10-7 corbaloc - basic

```
corbaloc::myhost
```

Example 10-8 corbaloc - cell's name space root via specific server

```
corbaloc:iiop:1.2@myhost.raleigh.ibm.com:9344/NameServiceCellRoot
```

Example 10-9 corbaloc - server name space root with fault tolerance

```
corbaloc::myhost1:9333,:myhost2:9333,:myhost2:9334/NameServiceServerRoot
```

Note: corbaloc URLs are usually used for the provider URL when retrieving an InitialContext.

corbaname

The corbaloc form of the CORBA 2.3 URL has the following syntax.

```
corbaname:<protocol>:<addresslist>/<key>#<INS string-formatted-name>
```

Table 10-2 corbaname options

Setting	Description
protocol	The protocol used for the communication. Currently, the only valid value is iiop.
addresslist	List of one or more addresses (host name and port number). The addresses are separated by commas, and each address has a colon prefix.
key	Type of root to access. See Table 10-5 on page 424 for details.
`<INS string-formatted-name>`	Fully qualified path to entry under the specific root context.

The following examples illustrate how the corbaname URL can range from simple to complex, depending upon whether fault tolerance (request retry with second, third, etc. server) is required.

Example 10-10 corbaname - fully qualified name access

```
corbaname::myhost:9333#cell/nodes/node1/servers/server5/someEjb
```

Example 10-11 corbaname - object accessed via specific server root

```
corbaname::myhost:9333/NameServiceServerRoot/someEjb
```

Note: corbaname URLs are usually used when performing a direct URL lookup using a previously obtained InitialContext, for example ic.lookup("urlstring").

10.4 Distributed CosNaming

One of the advantages of the new distributed nature of CosNaming in IBM WebSphere Application Server is that it removes the bottleneck of having a single name server for all naming lookups.

In WebSphere Application Server V5, each WebSphere process (Deployment Manager, node agent and application server) hosts its own ORB, NameService and local name space. As such, lookups are made by accessing the NameService in the most convenient process; they are not bottlenecked through a single server process in the cell.

Note: The JMS server process does not host an ORB, NameService or a local name space. Any references to JMS objects running in this process are actually registered in the local name spaces of the application server(s) that access the JMS objects.

To enable the new distributed naming architecture, the following major changes have been made to CosNaming:

- The IBM WebSphere Application Server Naming architecture uses CORBA CosNaming as its foundation.
- The CosNaming architecture has been changed to support a distributed and federated collection of CosNaming servers:
  - Each Deployment Manager, node agent, and application server is a CosNaming server. Each server is responsible for managing the names of the objects bound locally.
  - Objects are bound into the local context.
  - Lookups start in the local process and end in the process where the target object is located.
  - This reduces the dependency of the WebSphere Application Server network on a single name server for all lookups.
- A single logical name space exists across the cell. The separate name spaces of each server process are linked/federated via context links in the cell name space. It is possible to navigate to specific subcontexts, as every server cluster and non-clustered server has its own context stored in a level of the cell name space.
- The contents of the federated name space are mostly transient, built from configuration data read from the XML configuration files on the startup of each server process.

- Persistent roots are provided at the cell and node level of the name space in order to provide locations where objects can be persistently bound. These bindings are persisted to XML files on the file system.
- Each separate server process has its own bootstrap port, thereby reducing bottlenecks.

10.5 Configured bindings

The configured bindings feature allows objects to be added to the name space using the admin interfaces (wsadmin or administrative console).

The IBM WebSphere Application Server Naming component provides new functionality by which an administrator can explicitly add bindings to the cell name space without having to write code. This allows an administrator to configure an “alias” in a persistent name space that refers to an actual reference in one of the local name spaces. This provides an additional level of indirection for V5 names.

The functionality is useful in the following areas:

- Federation of name spaces

  As long as it is CORBA 2.3 compliant (supporting INS), the name space of other WebSphere Application Server V5 cells, WebSphere Application Server V4 administrative domains, third-party application servers and even CORBA servers can be federated into the cell’s name space.

- Interoperability with WebSphere Application Server V4

  The default context of WebSphere Application Server V4 clients is the global (legacy) context. However, WebSphere Application Server V5 processes bind their objects in local transient name spaces. Therefore, in order for WebSphere Application Server V4 clients to lookup and access objects in WebSphere Application Server V5 without requiring changes to the client, requires that the WebSphere Application Server V5 object be bound into the legacy name space accessible to the client. This is where configured bindings come in. An alias can be configured into the legacy name space. When used by the WebSphere Application Server V4 client, the client will be transparently redirected to the real object reference in one of the cell’s local name spaces.

10.5.1 Types of object

The following types of object can be bound using configured bindings:

- EJB hosted by a server in the cell

  The configured binding identifies an EJB home based on its configured JNDI name and the server in which it is deployed.

  A possible use of this is to put a binding for an EJB into the cell-scoped name space so that a lookup can be done without knowledge about the server in which the EJB is deployed. This mechanism is useful for allowing WebSphere Application Server V4 clients to look up WebSphere Application Server V5 EJBs without having to redeploy.

- CORBA object

  The configured binding identifies a CORBA object bound somewhere in this or another name space by using a corbaname URL string. Included is also an indicator of whether the object is a CosNaming NamingContext, in which case the binding is actually a federated link from one name space to another.

- JNDI name

  The configured binding identifies a provider URL and a JNDI name that can be used to look up an object. This can be used to reference a resource or (other Java serialized object) bound elsewhere in this name space or another name space.

- String constant

  Can be used to bind environment data into the name space.

10.5.2 Types of binding reference

There are a number of different types of references that can be specified for configured bindings. Valid types are summarized in Table 10-3.

Table 10-3 Types of binding reference

Binding type	Required Settings
EJB (EjbNameSpaceBinding)	1. Name in name space is relative to configured root. 2. JNDI name of EJB. 3. Server or server cluster where EJB is deployed.

Binding type	Required Settings
CORBA (CorbaObjectNameSpaceBinding)	1. Name in name space is relative to configured root. 2. corbaname URL. 3. Indicator if the target object is a federated context object, or a leaf node object
Indirect (IndirectLookupNameSpaceBinding)	1. Name in name space is relative to configured root. 2. Provider URL. 3. JNDI name of object.
String (StringNameSpaceBinding)	1. Name in name space is relative to configured root. 2. Constant string value.

The configured bindings can be relative to one of the following context roots:

- Server root
- Node persistent root
- Cell persistent root

10.6 Initial contexts

In WebSphere, an initial context for a name server is associated with a bootstrap host and bootstrap port. These combined values can be viewed as the address of the name server that owns the initial context. To get an initial context, the bootstrap host and port for the initial context's name server must be known.

JNDI clients should assume the correct environment is already configured, so there is no need to explicitly set property values and pass them to the InitialContext constructor.

However, a JNDI client may need to access a name space other than the one identified in its environment. In this case, it is necessary to explicitly set the javax.naming.provider.url (provider URL) property used by the InitialContext constructor. A provider URL contains bootstrap server information that the initial context factory can use to obtain an initial context. Any property values passed directly to the InitialContext constructor take precedence over settings of those same properties found elsewhere in the environment.

Two different provider URL forms can be used with WebSphere's initial context factory:

- CORBA object URL (new for J2EE 1.3)
- IIOP URL

CORBA object URLs are more flexible than IIOP URLs and are the recommended URL format to use. CORBA object URLs are part of the OMG CosNaming Interoperable Naming Specification. The IIOP URLs are the legacy JNDI format, but are still supported by the WebSphere initial context factory.

The following examples illustrate the use of these URLs.

Using a CORBA object URL

An example of using a corbaloc URL with a single address to obtain an initial context is shown in Example 10-12.

Example 10-12 Initial context using CORBA object URL

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "corbaloc:iiop:myhost.mycompany.com:2809");

Context initialContext = new InitialContext(env);
```

Using a CORBA object URL with multiple addresses

CORBA object URLs can contain more than one bootstrap server address. This feature can be used in WebSphere when attempting to obtain an initial context from a server cluster. The bootstrap server addresses for all servers in the cluster can be specified in the URL. The operation will succeed if at least one of the servers is running, eliminating a single point of failure.

Note: There is no guarantee of any particular order in which the address list will be processed. For example, the second bootstrap server address may be used to obtain the initial context even though the first bootstrap server in the list is available.

An example of using a corbaloc URL with multiple addresses to obtain an initial context is shown in Example 10-13 on page 422.

Example 10-13 Initial context using CORBA object URL with multiple addresses

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL,
"corbaloc::myhost1:2809,:myhost2:2809,:myhost3:2809");

Context initialContext = new InitialContext(env);
```

Using a CORBA object URL from a non-WebSphere JNDI

To access a WebSphere name server from a non-WebSphere environment, such that the WebSphere initial context factory is not used, a corbaloc URL must be used that has an object key of *NameServiceServerRoot* to identify the server root context.

The server root is where system artifacts such as EJB homes are bound. The default key of NameService can be used when fully qualified names are used for JNDI operations.

Example 10-14 shows a CORBA object type URL from a non-WebSphere JNDI implementation. It assumes full CORBA object URL support by the non-WebSphere JNDI implementation.

Example 10-14 Using a CORBA object URL from non-WebSphere JNDI

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable(); env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.somecompany.naming.TheirInitialContextFactory");
env.put(Context.PROVIDER_URL,
"corbaname:iiop:myhost.mycompany.com:2809/NameServiceServerRoot");

Context initialContext = new InitialContext(env);
```

Using an IIOP URL

The IIOP type of URL is a legacy format that is not as flexible as CORBA object URLs. However, URLs of this type are still supported by the WebSphere initial context factory.

Example 10-15 shows an IIOP type URL as the provider URL.

Example 10-15 Initial context using an IIOP URL

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "iiop://myhost.mycompany.com:2809");

Context initialContext = new InitialContext(env);
```

10.6.1 Setting initial root context

Each server contains its own server root context, and when bootstrapping to a server, the server root is the default initial JNDI context. Most of the time, this is the desired initial context, since system artifacts such as EJB homes are bound at this point. However, other root contexts exist that may contain bindings of interest. It is possible to specify a provider URL to select other root contexts.

Note: The default is that the name will be resolved based upon the context associated with the server whose bootstrap the client is connected.

The initial root context can be selected using the following settings:

- CORBA object URL
- Name space root property

Default initial context

The default initial context depends on the type of client. Table 10-4 summarizes the different categories of clients and the corresponding default initial context.

Table 10-4 Default initial context versus client type

Client type	Description	Default initial context
WebSphere Application Server V5 JNDI	The JNDI interface is used by EJB applications to perform name space lookups. WebSphere clients by default use WebSphere's CosNaming JNDI plug-in implementation.	Server root

Client type	Description	Default initial context
WebSphere JNDI prior to V5	WebSphere clients running in releases prior to V5 by default use WebSphere's V4 CosNaming JNDI plug-in implementation.	Cell persistent root (legacy root)
Other JNDI	Some applications may perform name space lookups with a non-WebSphere CosNaming JNDI plug-in implementation.	Cell root
CORBA		Cell root

Selecting initial root context with a CORBA object URL

There are several object keys registered with the bootstrap server that can be used to select the root context to be used as the initial context. To select a particular root context with a CORBA object URL object key, set the object key to the corresponding value. The default object key is NameService. Using JNDI, this will yield the server root context.

Table 10-5 lists the different root contexts and their corresponding object key.

Table 10-5 CORBA object URL root context values

Root context	CORBA object URL object key	Description
Server root	NameServiceServerRoot	Server root for the server being accessed.
Cell persistent root	NameServiceCellPersistentRoot	The persistent cell root for the server being accessed.
Cell root	NameServiceCellRoot	The cell root for the server being accessed.
Node root	NameServiceNodeRoot	The node root for the server being accessed.

Note: The name server running in the Deployment Manager process has no node root registered under the NameServiceNodeRoot key, because there is no node agent or application servers running in its node.

Example 10-16 on page 425 shows the use of a corbaloc URL with the object key set to select the cell persistent root context as the initial context.

Example 10-16 Select cell persistent root context using corbaloc URL

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL,
"corbaloc:iiop:myhost.mycompany.com:2809/NameServiceCellPersistentRoot");

Context initialContext = new InitialContext(env);
```

Selecting initial root context with name space root property

The initial root context can be selected by passing a name space root property setting to the InitialContext constructor. Generally, the object key setting described above is sufficient. Sometimes a property setting might be preferable.

For example, the root context property can be set on the Java invocation to make it transparent to the application which server root is being used as the initial context. The default server root property setting is *defaultroot*, which will yield the server root context.

Tip: If a simple name is used, the root context that will be assumed can be set by passing the `com.ibm.websphere.naming.namespaceroot` property to InitialContext.

Table 10-6 Name space root values

Root context	CORBA object URL object key
Server root	bootstrapserverroot
Cell persistent root	cellpersistentroot
Cell root	cellroot
Node root	bootstrapnoderoot

The name space root property is used to select the default root context only if the provider URL does not contain an object key or contains the object key, *NameService*. Otherwise, the property is ignored.

Example 10-17 on page 426 shows use of the name space root property to select the cell persistent root context as the initial context.

Tip: WebSphere makes available constants that can be used instead of hard-coding the property name and value, for example

```
env.put(PROPS.NAME_SPACE_ROOT, PROPS.NAME_SPACE_ROOT_CELL_PERSISTENT);
```

Example 10-17 Use of name space root property to select cell persistent root context

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
import com.ibm.websphere.naming.PROPS;

Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "corbaloc:iiop:myhost.mycompany.com:2809");
env.put(PROPS.NAME_SPACE_ROOT, PROPS.NAME_SPACE_ROOT_CELL_PERSISTENT);

Context initialContext = new InitialContext(env);
```

10.7 Federation of name spaces

Federating name spaces involves binding contexts from one name space into another name space. In a WebSphere Application Server V5 name space, federated bindings can be created with the following restrictions:

- Federation is limited to CosNaming name servers. A WebSphere name server is a Common Object Request Broker Architecture (CORBA) CosNaming implementation.

  Federated bindings to other CosNaming contexts can be created, but bindings to LDAP name server implementation contexts cannot.

- If JNDI is used to federate the name space, the WebSphere initial context factory must be used to obtain the reference to the federated context. If any other initial context factory implementation is used, the binding may not be able to be created, or the level of transparency may be reduced.
- A federated binding to a non-WebSphere naming context has the following functional limitations:
  - JNDI operations are restricted to the use of CORBA objects. For example, EJB homes can be looked up, but non-CORBA objects such as data sources cannot.
  - JNDI caching is not supported for non-WebSphere name spaces. This only affects the performance of lookup operations.

- Do not federate two WebSphere single server name spaces. If this is done, incorrect behavior may result. If federation of WebSphere name spaces is required, then servers running under IBM WebSphere Application Server Network Deployment need to be used.

As an example, assume that a name space, Namespace 1, contains a context under the name “a/b.” Also assume that a second name space, Namespace 2, contains a context under the name “x/y.” If context “x/y” in Namespace 2 is bound into context “a/b” in Namespace 1 under the name “f2,” the two name spaces are federated. Binding “f2” is a federated binding because the context associated with that binding comes from another name space. As shown in Figure 10-3, from Namespace 1, a lookup of the name “a/b/f2” would return the context bound under the name “x/y” in Namespace 2. Furthermore, if context “x/y” contained an EJB home bound under the name “ejb1,” the EJB home could be looked up from Namespace1 with the lookup name “a/b/f2/ejb1.” Notice that the name crosses name spaces. This fact is transparent to the naming client.

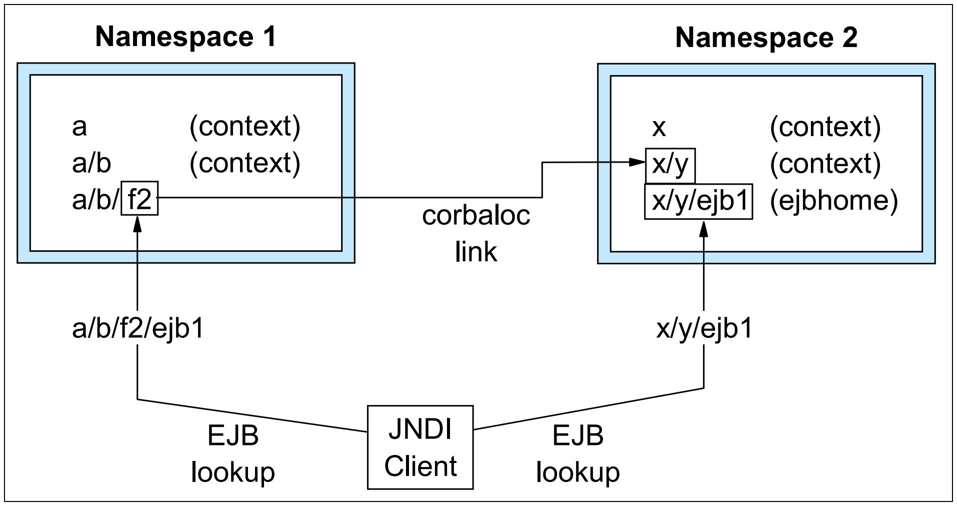


Figure 10-3 JNDI access using federated name spaces

10.8 Interoperability

WebSphere Application Server V5 provides support for interoperating with previous releases of WebSphere and with non-WebSphere JNDI clients:

- EJB clients running on WebSphere V3.5.x or V4.0.x, accessing EJB applications running on WebSphere Application Server V5.
- EJB clients running on WebSphere Application Server V5, accessing EJB applications running on WebSphere V3.5.x or V4.0.x servers.

- EJB clients running in an environment other than WebSphere, accessing EJB applications running on WebSphere Application Server V5 servers.

10.8.1 WebSphere V3.5/V4.0 EJB clients

Applications migrated from previous WebSphere releases may still have clients running in a previous release. The default initial JNDI context for EJB clients running on previous versions of WebSphere is the cell persistent root (legacy root). However, the home for an EJB deployed in V5.0 is bound to the server root context. In order for the EJB lookup name for down-level clients to remain unchanged, configure a binding for the EJB home under the cell persistent root.

The following options are available for enabling interoperability with WebSphere Application Server V4 clients:

- Set the client's default initial context to legacyRoot. This option is equivalent to the cell persistent root of WebSphere Application Server V5.
- Redeploy the clients using the Assembly Toolkit so that the JNDI names can be fixed to reflect the actual fully qualified names in the WebSphere Application Server V5 name space.
- Use aliases for the names the clients look up. These transparently redirect to the correct object in the WebSphere Application Server V5 name space. This option uses configured bindings.

Important: In order to interoperate with WebSphere Application Server V5, EJB clients running on WebSphere V3.5.x must be running at V3.5.5, or at V3.5.3/V3.5.4 with fix PQ51387 (or an updated fix) installed.

Options for EJB lookup

The following options exist for supporting EJB lookup from a WebSphere Application Server V4 client to a WebSphere Application Server V5 hosted EJB:

1. Redeploy the WebSphere Application Server V4 client.

   The <ejb-ref> needs to be updated to reflect the WebSphere Application Server V5 compatible JNDI name.

2. In WebSphere Application Server V5, configure EjbNameSpaceBinding:
   a. Use the same JNDI name as looked up by the WebSphere Application Server V4 client.
   b. Identify the JNDI name and server (or cluster) of the target EJB.
   c. Configure the binding in the cell persistent root.

Options for resources bound in external name space

1. Redeploy the WebSphere Application Server V4 client.

   The <resource-ref> needs to be updated to reflect the WebSphere Application Server V5 compatible JNDI name.

2. In WebSphere Application Server V5, run the program to bind the resource into the WebSphere Application Server V5 cell persistent root.
3. In WebSphere Application Server V5, configure IndirectLookupNameSpaceBinding:
   a. Use the same JNDI name as looked up by the WebSphere Application Server V4 client.
   b. Specify the provider URL and JNDI name of the name space where the resource is already bound (a WebSphere Application Server V4 name space).
   c. Configure the binding in the cell persistent root.

10.8.2 WebSphere V3.5/V4.0 servers

The default initial context for a WebSphere V3.5 or V4.0 server is the correct context. WebSphere Application Server V5 clients simply look up the JNDI name under which the EJB home is bound.

Important: To enable WebSphere Application Server V5 clients to access Version 3.5.x and V4.0.x servers, the down-level installations must have the fix PQ60074 (or an updated fix) installed.

10.8.3 EJB clients hosted by non-WebSphere environment

When an EJB application running in WebSphere Application Server V5 is accessed by a non-WebSphere EJB client, the JNDI initial context factory is presumed to be a non-WebSphere implementation. In this case, the default initial context will be the cell root. If the JNDI service provider being used supports CORBA object URLs, the following corbaname format can be used to look up the EJB home.

Example 10-18 Use corbaname format for EJB home lookup

```
initialContext.lookup("corbaname:iiop:myHost:2809#cell/clusters/myCluster/myEJB
");
```

According to the URL in Example 10-18 on page 429, the bootstrap host and server (node agent) port are “myHost” and 2809. The EJB is installed in a server cluster named “myCluster”. The EJB is bound in that cluster under the name “myEJB”.

Note: The server name could just as well be the name of a non-clustered server. This form of lookup will work:

- With any name server bootstrap host and port configured in the same cell.
- If the bootstrap host and port belongs to a member of the cluster itself.

To avoid a single point of failure, the bootstrap server host and port for each cluster member could be listed in the URL as shown in Example 10-19.

Example 10-19 Use corbaname format with multiple addresses for EJB home lookup

```
initialContext.lookup("corbaname:iiop:host1:9810,host2:9810#cell/clusters/myClu
ster/myEJB");
```

The name prefix `cell/clusters/<clustername>/` is not necessary if bootstrapping to the cluster itself, but it will always work. The prefix is required, however, when looking up EJBs in other clusters. The server binding for the prefix used to access another cluster is implemented in a way that avoids a single point of failure during a lookup.

If the JNDI initial context factory being used does not support CORBA object URLs, the initial context can be obtained from the server, and the lookup can be performed on the initial context as shown in Example 10-20.

Example 10-20 Use corbaname format with multiple addresses for EJB home lookup

```
Hashtable env = new Hashtable();
env.put(CONTEXT.PROVIDER_URL, "iiop://myHost:2809");
Context ic = new InitialContext(env);
Object o = ic.lookup("cell/clusters/myCluster/myEJB");
```

This form of lookup will work from any server in the same cell as the EJB home being looked up. However, this approach does not allow multiple hosts and ports to be specified in the provider URL and hence does not incorporate the availability advantages of a corbaloc or corbaname URL with multiple hosts and ports belonging to the server cluster members.

10.9 Examples

The following examples highlight a number of different server topologies and the affect the topologies have on the use of the Naming Service:

- Single server.
- Single server with a non-default port.
- Two single servers on the same box.
- Two Network Deployment application servers on the same box.
- WebSphere Application Server V4 client.

10.9.1 Single server

In the single-server environment, the naming functionality works in exactly the same way as in WebSphere Application Server V4. There is only one server and only one root context and therefore no ambiguity in the location of a named object. This is illustrated by the example in Figure 10-4.

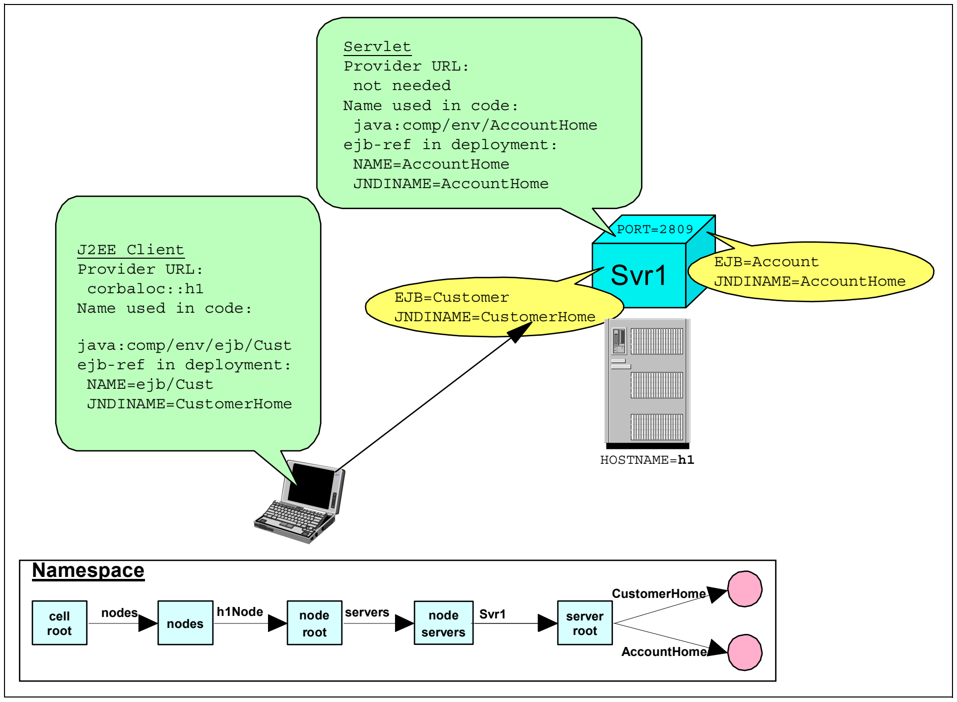


Figure 10-4 Single server

By accessing the server root directly, the client (J2EE or servlet) does not need to traverse the cell name space (cell root -> servers -> server root -> object).

Note: Even in a single-server case, clients can still use the fully qualified JNDI name to look up an object from the cell root. This would remove any dependency on the particular topology in use, at the cost of a small performance degradation due to the extra lookup to retrieve the server root from the cell name space.

In a single-server case, the server root acts as the default bootstrap, and therefore should be assigned port 2809. In this case, the provider URL used by clients external to the server process does not need to include a port number.

Tip: If the named object is looked up by a client running in the same process, then a provider URL (and hence corbaloc) does not need to be provided. By default the lookup will be performed against the local process' name space.

Table 10-7 illustrates the Provider URL settings required.

Table 10-7 Lookup settings required for a single server

Component	Provider URL	JNDI name
Servlet (same process	*Not needed*	CustomerHome
J2EE client (external process)	corbaloc::<hostname>	CustomerHome

10.9.2 Two single servers on the same box

When more than one instance of the application server runs on a single machine, then each server's bootstrap must be configured to run on a different port. In this case, you can have a J2EE component in one server looking up objects in the other server.

This is illustrated by the example in Figure 10-5 on page 433.

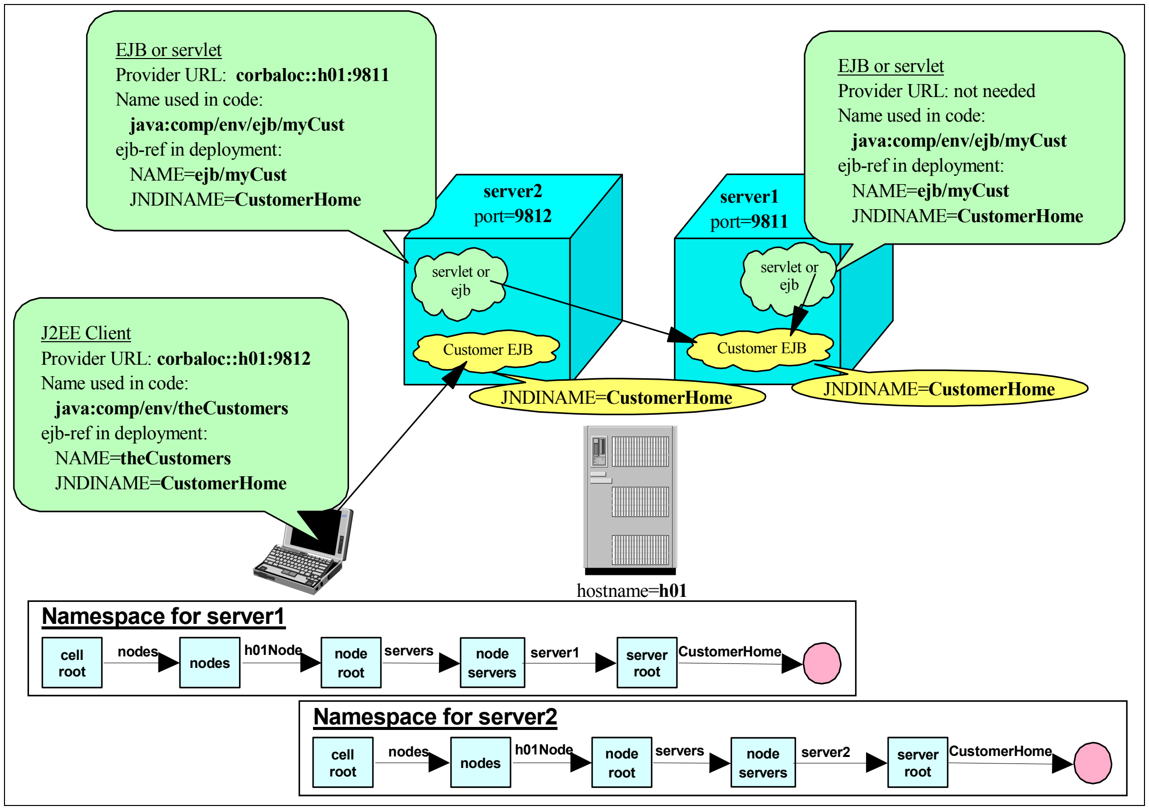


Figure 10-5 Two single servers on the same box

Since each of the application server name spaces are separate, the different objects can be registered in each under the same name (CustomerHome). There is no problem with name collision.

Note: The fully qualified JNDI name can be used to uniquely identify the name registered in one server from the name in another. For example, if objects are registered under the name CustomerHome on two servers, the specific name can be looked up using:

```
cell/nodes/<nodename>/servers/<server1>/CustomerHome
cell/nodes/<nodename>/servers/<server2>/CustomerHome
```

Table 10-8 on page 434 illustrates the Provider URL settings required.

Table 10-8 Lookup settings required for two single servers on same box

Component	Provider URL	JNDI name
Servlet (same process)	*Not needed*	CustomerHome
Servlet (external process)	corbaloc::<hostname>:<port#>	CustomerHome
J2EE client (external process)	corbaloc::<hostname>:<port#>	CustomerHome

10.9.3 Two Network Deployment application servers on the same box

The configuration becomes more complex when we move from an IBM WebSphere Application Server Base to a Network Deployment configuration. In this topology, there can be a number of separate application servers as well as a node agent process, all of which have a bootstrap port and host a local name space:

- Node agent

  The node agent is the default bootstrap for the box, and therefore has its bootstrap port configured on 2809.

- Application servers

  The application servers are not the default bootstrap, and therefore each is configured to use a non-default bootstrap port.

This is illustrated by the example in Figure 10-6 on page 435.

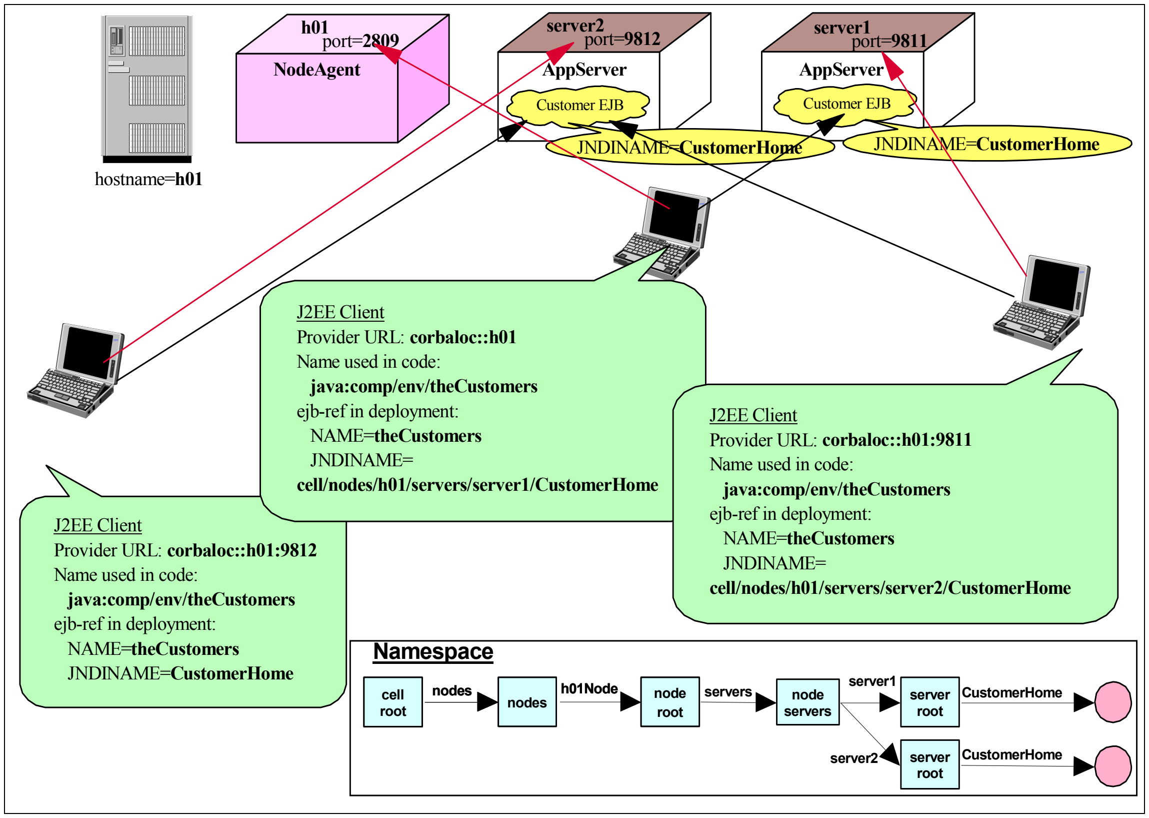


Figure 10-6 Two network deployment application servers on the same box

Unless a client specifies a specific application server in its provider URL, the lookup will be performed on the node agent. In order for the lookup to succeed, the bindings have to specify the fully qualified name of the object:

```
cell/nodes/<nodename>/servers/<servername>/<name of object>
```

That is, the client needs to specify where the object is located. This is a big difference from the behavior in WebSphere Application Server V4, where all named objects were registered in a single global name space.

Tip: When making use of server clusters for high availability in a Network Deployment configuration, bootstrap to a server cluster so that the initial context will have failover support. Lookups which resolve to other clusters from that bootstrap cluster also have failover support due to the name server implementation. The provider URL should have the bootstrap address of each cluster member to avoid a single point of failure when obtaining the initial context.

In Network Deployment configurations, choose a bootstrap server which is expected to have a stable bootstrap address. Consider designating a particular cluster, server, or node agent for client bootstrapping.

Table 10-9 illustrates the Provider URL settings required.

Table 10-9 Lookup settings required for two Network Deployment servers on the same box

Component	Provider URL	JNDI name
Servlet (same process)	*Not needed*	CustomerHome
Servlet (external process accessing local name space to access local object)	*Not needed*	cell/nodes/<nodename>/servers/<server2>/CustomerHome
Servlet (external process accessing other appserver's name space to access object on that appserver)	corbaloc::<appserver2 hostname>:<port#>	CustomerHome
	(or) Not needed	cell/persistent/CustomerHome2[1]
J2EE client (external process accessing appserver1 with object located on appserver1)	corbaloc::<appserver hostname>:<port#>	CustomerHome
	(or) Not needed	cell/persistent/CustomerHome1[1]
J2EE client (external process accessing node agent)	corbaloc::<node agent hostname>	cell/nodes/<nodename>/servers/<server2>/CustomerHome
	(or) Not needed	cell/persistent/CustomerHome2[1]

[1]Indirect JNDI references to the respective EJB must be manually configured in the cell/persistent name space.

Component	Provider URL	JNDI name
J2EE client (external process accessing appserver1 with object located on appserver2)	corbaloc::<appserver1 hostname>:<port#>	cell/nodes/<nodename>/servers/<server2>/CustomerHome
	(or) Not needed	cell/persistent/CustomerHome2[1]
[1] Indirect JNDI references to the respective EJB must be manually configured in the cell/persistent name space.		

10.9.4 WebSphere Application Server V4 client

In WebSphere Application Server V4, there is no need to specify a path to a named object, since all objects are registered in a single global name space. Although convenient, this causes naming conflicts since no two objects can be registered across all application servers with the same names.

The use of configured bindings (aliases) in the cell persistent root provides a mechanism by which the V4 style naming structure can be mapped to the fully qualified names of V5. This is illustrated by the example in Figure 10-7 on page 438.

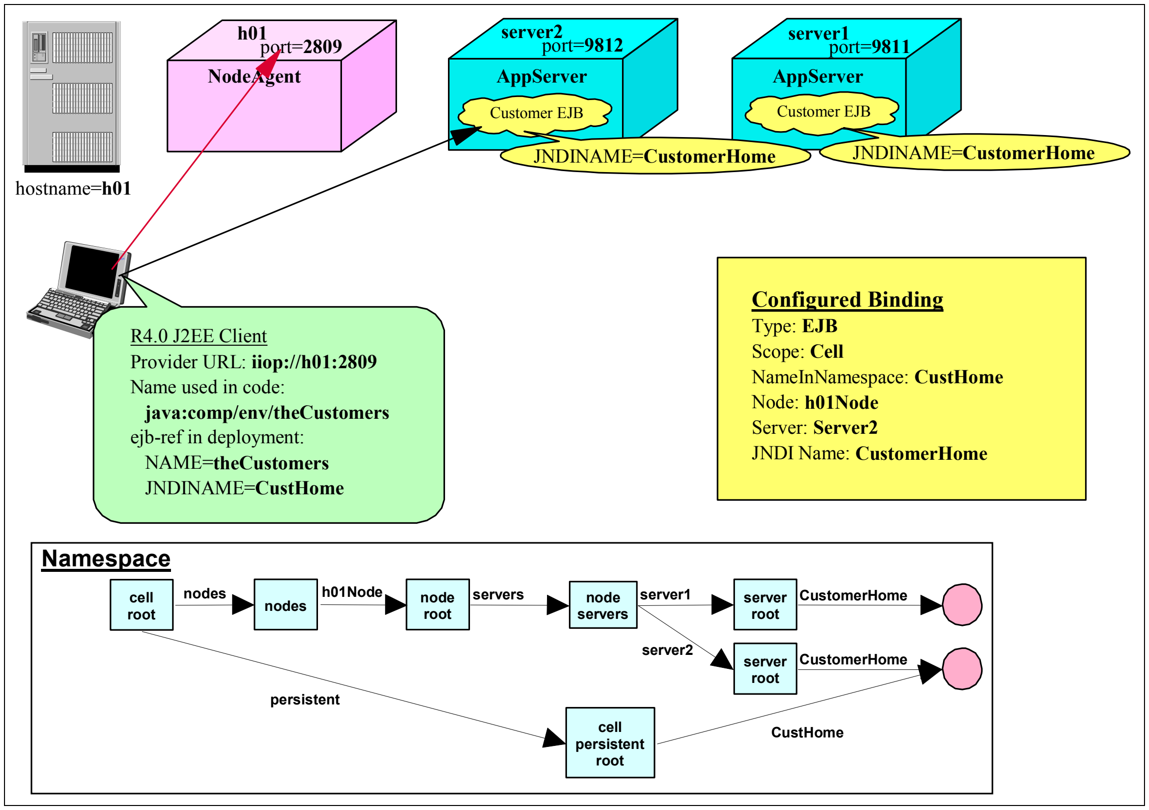


Figure 10-7 WebSphere Application Server V4 client

Table 10-10 illustrates the Provider URL settings required.

Table 10-10 Lookup settings required for WebSphere Application Server V4 client interoperability

Component	Provider URL	JNDI name
V4 client	iiop://<hostname>:2809	CustHome

CustHome is the name registered in the cell-level persistent root (legacy root) for cell/nodes/<nodename>/servers/<servername>/CustomerHome.

The WebSphere Application Server V4 client accesses the JNDI alias registered in the cell persistent root (legacyRoot) of the WebSphere Application Server V5 cell. The WebSphere Application Server V5 runtime *transparently* redirects the client to the actual JNDI entry located in a specific local name space hosted by one of the name servers of the cell.

10.10 Naming tools

IBM WebSphere Application Server provides the following tools for the support of the naming architecture.

10.10.1 dumpNameSpace

The **dumpNameSpace** command can be run against any bootstrap port to get a listing of the names bound with that provider URL.

The output of the command:

- Does not present a full logical view of the name space.
- Shows CORBA URLs where the name space transitions to another server.

The tool indicates that certain names point to contexts that are external to the current server (and its name space). The links show the transitions necessary to perform a lookup from one name space to another.

Tip: An invocation of the **dumpNameSpace** command cannot generate a dump of the entire name space, only the objects bound to the bootstrap server and links to other local name spaces that compose the federated name space. Use the correct host name and port number for the server to be dumped.

Syntax

```
dumpNameSpace [options]
```

All arguments are optional.

Table 10-11 Options for dumpNameSpace

Option	Description
-host <hostname>	Host name of bootstrap server. If not defined, then the default is localhost.
-port <portnumber>	Bootstrap server port number. If not defined, then the default is 2809.
-factory <factory>	Initial context factory to be used to get initial context. The default of com.ibm.websphere.naming.WsnInitialContextFactory should be ok for most use.

Option	Description
-root [cell | server | node | host | legacy | tree | default]	*For WebSphere V5.0 or later:* cell: dumpNameSpace default. Dump the tree starting at the cell root context. server: Dump the tree starting at the server root context. node: Dump the tree starting at the node root context. (Synonymous with "host") *For WebSphere V4.0 or later:* legacy: dumpNameSpace default. Dump the tree starting at the legacy root context. host: Dump the tree starting at the bootstrap host root context (Synonymous with "node") tree: Dump the tree starting at the tree root context. *For all WebSphere and other name servers:* default: Dump the tree starting at the initial context which JNDI returns by default for that server type. This is the only -root choice that is compatible with WebSphere servers prior to V4.0 and with non-WebSphere name servers.
-url <url>	The value for the java.naming.provider.url property used to get the initial JNDI context. This option can be used in place of the -host, -port, and -root options. If the -url option is specified, the -host, -port, and -root options are ignored.
-startAt <context>	The path from the requested root context to the top level context where the dump should begin. Recursively dumps subcontexts below this point. Defaults to empty string, that is, root context requested with the -root option.
-format <format>	“jndi”: Display name components as atomic strings. “ins”: Display name components parsed per INS rules (id.kind). The default format is “jndi”.

Option	Description
-report <length>	short: Dumps the binding name and bound object type, which is essentially what JNDI Context.list() provides. long: Dumps the binding name, bound object type, local object type, and string representation of the local object (that is, IORs, string values, etc., are printed). The default report option is "short".
-traceString <tracespec>	Trace string of the same format used with servers, with output going to the file "DumpNameSpaceTrace.out".
-help	Prints a usage statement.
-?	Prints a usage statement.

Finding the bootstrap address

The bootstrap address for node agents, servers, and the cell are all found under the End Points option of the Additional Properties pane on their respective configuration page.

- For application servers, click **Servers -> Application Servers**. Select the server to open the configuration. Select **End Points**, then **BOOTSTRAP ADDRESS**.
- For node agents, click **System Administration -> Node Agents**. Select the node agent to open the configuration. Select **End Points**, then **BOOTSTRAP ADDRESS**.
- For the cell, click **System Administration -> Deployment Manager**. Select **End Points**, then **BOOTSTRAP ADDRESS**.

Example 10-21 dumpNameSpace usage

```
$ cd c:\ibm\was50\AppServer\bin

Get help on options:
$ dumpNameSpace -?

Dump server on localhost:2809 from cell root:
$ dumpNameSpace

Dump server on localhost:2806 from cell root:
$ dumpNameSpace -port 2806

Dump server on yourhost:2811 from cell root:
```

```
$ dumpNameSpace -port 2811 -host yourhost

Dump server on localhost:2809 from server root:
$ dumpNameSpace -root server‘

Dump server at corbaloc
dumpNameSpace -url corbaloc:iiop:yourhost:901
```

10.11 Configuration

This section tells you how to configure a name binding for an enterprise bean, a CORBA CosNaming naming context or CORBA leaf node object, an object that can be looked up using JNDI, or a constant string value using the administrative console.

10.11.1 Name space bindings

The configured bindings feature allows objects to be added to the name space using the administrative console. An administrator can now explicitly add bindings to the cell name space without having to write code. This allows an administrator to configure an “alias” in a persistent name space that refers to an actual reference in one of the local name spaces.

Name space bindings can be created for the following four object types:

- String
- EJB
- CORBA
- Indirect

As an example, let’s look at Figure 10-7 on page 438. In this scenario an alias is configured to allow an application using the WebSphere V4 naming style to access an EJB while running on WebSphere V5. Because the V4 application code does not specify a path to the named object, a binding is added to the cell persistent root (legacy root) to redirect the client to the JNDI entry in the local name space.

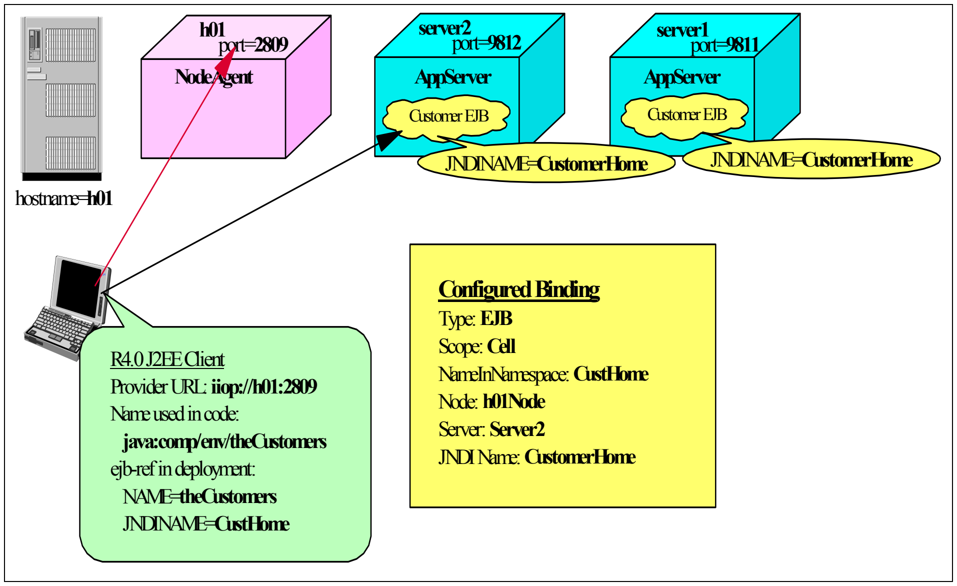
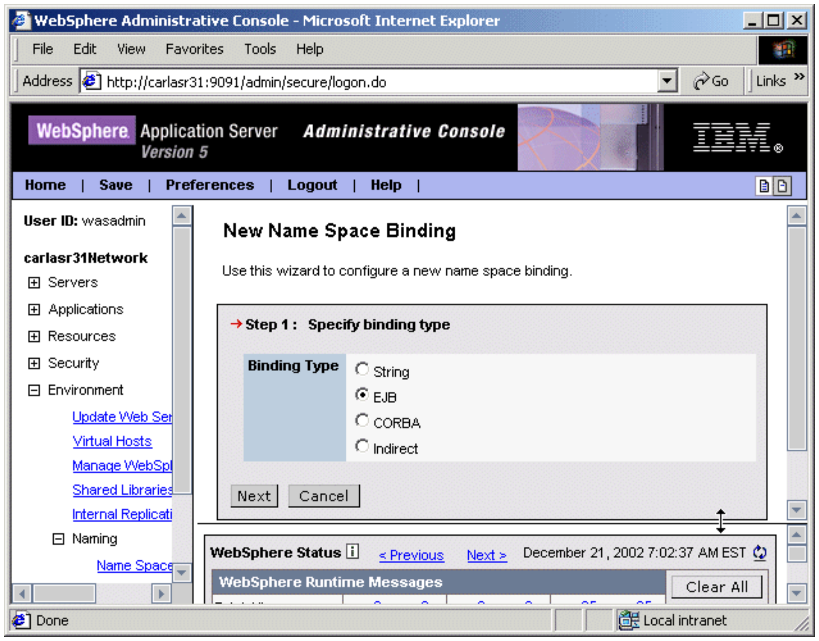


Figure 10-8 WebSphere Application Server V4 client

To create the binding:

1. Select **Environment -> Naming -> Name Space Bindings**.
2. Set the scope to `cell`.
3. Click **New**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: wasadmin
carlasr31Network
Servers
Applications
Resources
Security
Environment
Update Web Ser
Virtual Hosts
Manage WebSpl
Shared Libraries
Internal Replicati
Naming
Name Space
WebSphere Status < Previous Next > December 21, 2002 7:02:37 AM EST
WebSphere Runtime Messages
Clear All
Done
Local intranet

New Name Space Binding

Use this wizard to configure a new name space binding.

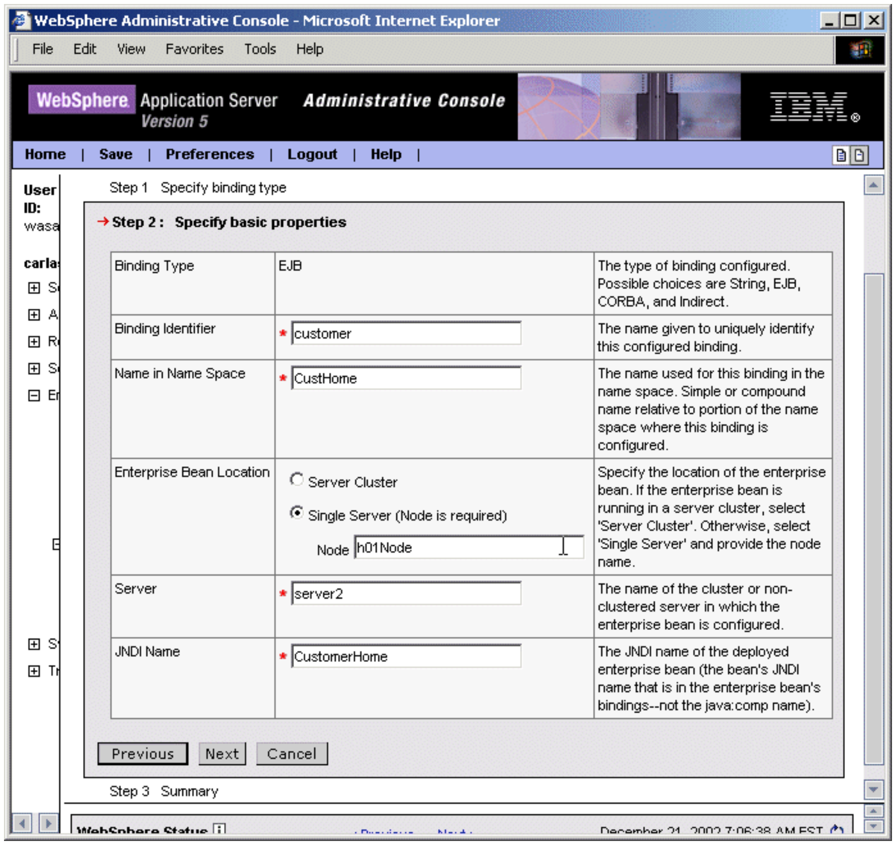
Step 1 : Specify binding type

Binding Type: String [✓ EJB] CORBA Indirect

Next Cancel

Figure 10-9 Name space binding

4. Choose **EJB** and click **Next.**
5. Enter the values as shown in Figure 10-10 on page 445.
   – Binding identifier: A unique identifier for the binding.
   – Name in Name Space: Matches the JNDI name used in the application code.
   – Enterprise Bean Location: The cluster or node where the EJB resides.
   – Server: Name of the server where the EJB resides.
   – JNDI Name: The JNDI name of the deployed EJB. This is the name in the enterprise beans bindings, not the java:comp name.


Step 1 Specify binding type

→ **Step 2 : Specify basic properties**

Binding Type	EJB	The type of binding configured. Possible choices are String, EJB, CORBA, and Indirect.
Binding Identifier	* customer	The name given to uniquely identify this configured binding.
Name in Name Space	* CustHome	The name used for this binding in the name space. Simple or compound name relative to portion of the name space where this binding is configured.
Enterprise Bean Location	○ Server Cluster ● Single Server (Node is required) Node h01Node	Specify the location of the enterprise bean. If the enterprise bean is running in a server cluster, select 'Server Cluster'. Otherwise, select 'Single Server' and provide the node name.
Server	* server2	The name of the cluster or non-clustered server in which the enterprise bean is configured.
JNDI Name	* CustomerHome	The JNDI name of the deployed enterprise bean (the bean's JNDI name that is in the enterprise bean's bindings--not the java:comp name).

Step 3 Summary

Figure 10-10 Defining an EJB name space binding

6. Click **Next**.
7. Click **Finish**.

Note: Name space bindings can be configured at the cell, node, and server scope:

- Bindings configured at the cell scope will be included in the local runtime name space of all application servers in that cell.
- Bindings configured at the node scope will be included in the local runtime name space of all application servers in that node.
- Bindings configured at the server scope will be included in the local runtime name space of only that application server.

10.11.2 CORBA naming service users and groups

The J2EE role-based authorization concept has been extended to protect the WebSphere CosNaming service. CosNaming security offers increased granularity of security control over CosNaming functions, which affect the content of the WebSphere name space. There are generally two ways in which client programs will make a CosNaming call. The first is through the JNDI interfaces. The second is CORBA clients invoking CosNaming methods directly.

Note: The authorization policy is only enforced when global security is enabled. Before enabling global security, you should design your entire security solution. See *IBM WebSphere V5.0 Security Handbook,* SG24-6573 for information on designing and implementing WebSphere security.

Authorization is done based on users and/or groups of users defined to the active user registry.

The authorization can be done by assigning an authority level to one of the following:

- User
- Group
- ALL_AUTHENTICATED (special subject that acts as a group): Any user that authenticates (enters a valid user ID and password).
- EVERYONE (special subject that acts as a group): All users are authorized. No authentication is necessary.

The roles now have authority level from low to high as follows:

- **CosNamingRead**. Users who have been assigned the CosNamingRead role will be allowed to do queries of the WebSphere Name Space, such as through the JNDI lookup method. The special subject “Everyone” is the default policy for this role.
- **CosNamingWrite**. Users who have been assigned the CosNamingWrite role will be allowed to do write operations such as JNDI bind, rebind, or unbind, plus CosNamingRead operations. The special subject “All_Authenticated” is the default policy for this role.
- **CosNamingCreate**. Users who have been assigned the CosNamingCreate role will be allowed to create new objects in the Name Space through such operations as JNDI createSubcontext, plus CosNamingWrite operations. The special subject “All_Authenticated” is the default policy for this role.
- **CosNamingDelete**. Users who have been assigned CosNamingDelete role will be able to destroy objects in the Name Space, for example using the JNDI destroySubcontext method, as well as CosNamingCreate operations.

By default, you have the following:

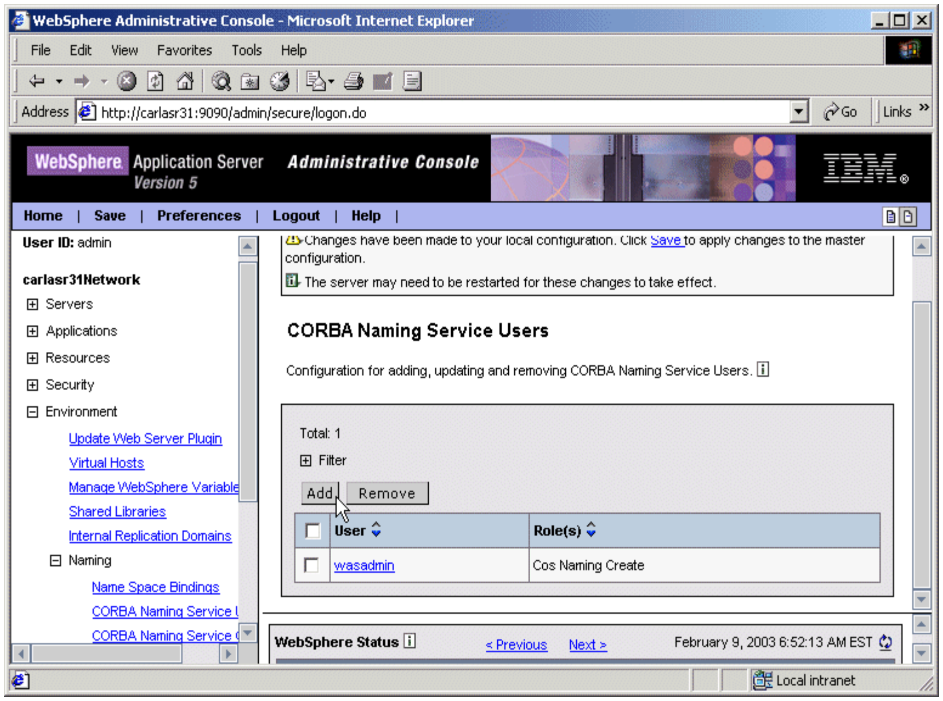
- The ALL_AUTHENTICATED group has the following role privileges: CosNamingRead, CosNamingWrite, CosNamingCreate, and CosNamingDelete.
- The EVERYONE group has CosNamingRead privileges only.

Working with the CORBA naming service authorization is straightforward.

Working with CORBA naming service users

To work with users:

1. Select **Environment -> Naming -> CORBA Naming Service Users**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Network
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variable
Shared Libraries
Internal Replication Domains
Naming
Name Space Bindings
CORBA Naming Service U
CORBA Naming Service G
Add
Remove
WebSphere Status
< Previous
Next >
February 9, 2003 6:52:13 AM EST
Local intranet

Changes have been made to your local configuration. Click Save to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

CORBA Naming Service Users

Configuration for adding, updating and removing CORBA Naming Service Users.

Total: 1

Filter

	User	Role(s)
☐	wasadmin	Cos Naming Create

Figure 10-11 Add CORBA naming service users

2. Click **Add**.

   Enter a user ID (case-sensitive) and select an authorization level. The user must be a valid user in the active user registry. If you have not activated global security, the local operating system user registry will be used. Remember, before these settings take effect, you will have to enable and configure WebSphere global security.

To specify multiple roles, hold the Ctrl key while you click the applicable roles.

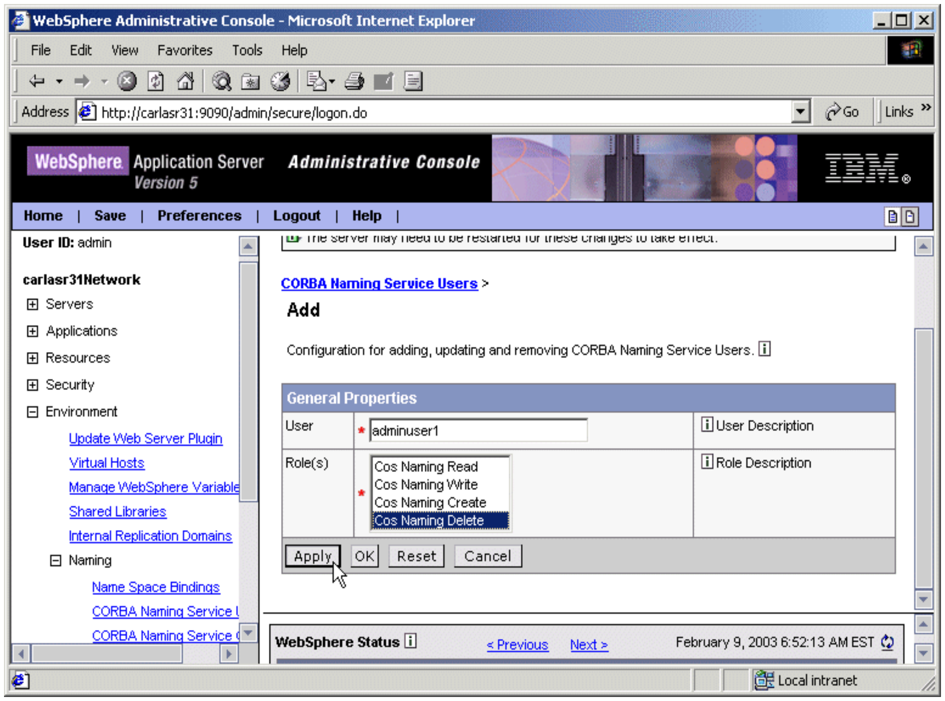

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Network
Servers
Applications
Resources
Security
Environment
Update Web Server Plugin
Virtual Hosts
Manage WebSphere Variable
Shared Libraries
Internal Replication Domains
Naming
Name Space Bindings
CORBA Naming Service U
CORBA Naming Service G
The server may need to be restarted for these changes to take effect.
CORBA Naming Service Users >
Add
Configuration for adding, updating and removing CORBA Naming Service Users.
General Properties
User
adminuser1
User Description
Role(s)
Cos Naming Read
Cos Naming Write
Cos Naming Create
Cos Naming Delete
Role Description
Apply
OK
Reset
Cancel
WebSphere Status
< Previous
Next >
February 9, 2003 6:52:13 AM EST
Local intranet

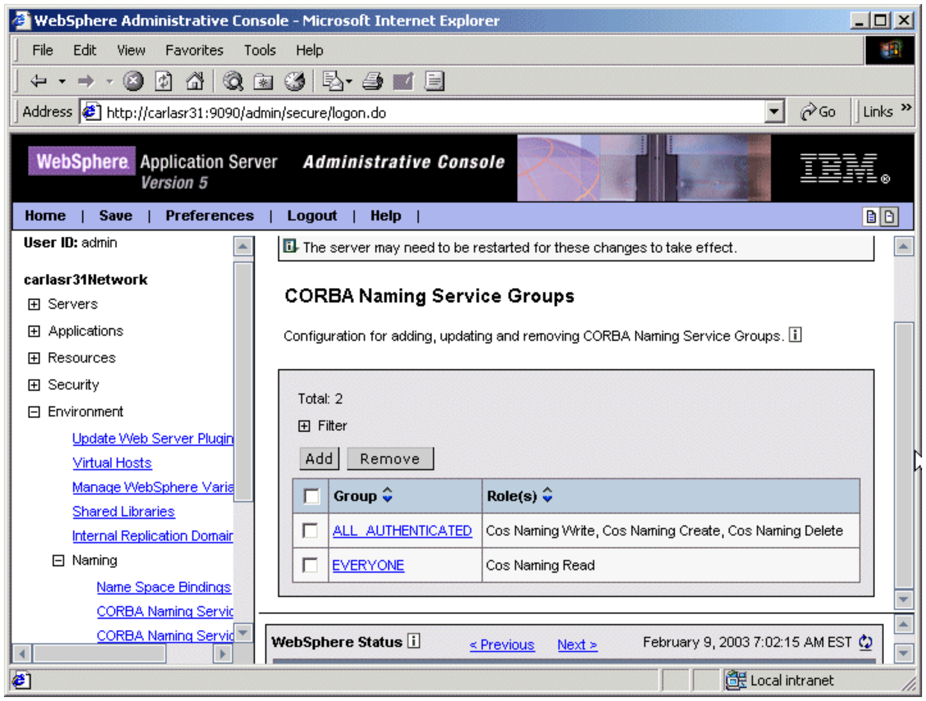
Figure 10-12 Assign an authorization level

3. Click **Apply**.
4. Click **OK**.

Working with CORBA naming service groups

To work with groups:

1. Select **Environment -> Naming -> CORBA Naming Service Groups**. You will see that the default settings are defined. Figure 10-13 on page 449 shows the initial settings.



The server may need to be restarted for these changes to take effect.

CORBA Naming Service Groups

Configuration for adding, updating and removing CORBA Naming Service Groups.

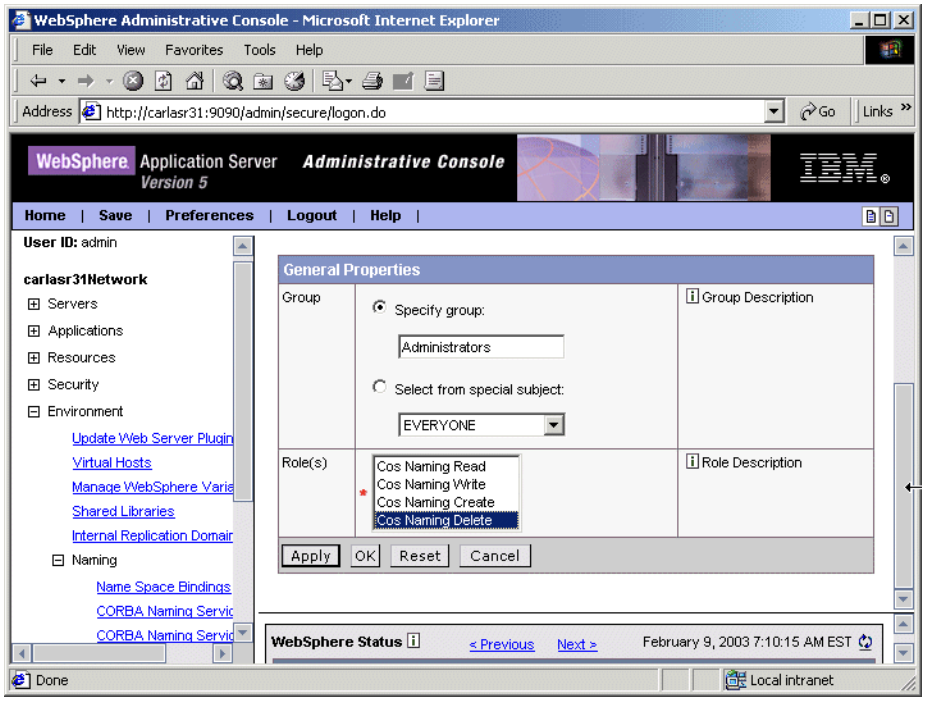
	Group	Role(s)
☐	ALL_AUTHENTICATED	Cos Naming Write, Cos Naming Create, Cos Naming Delete
☐	EVERYONE	Cos Naming Read

February 9, 2003 7:02:15 AM EST

Figure 10-13 Default settings for CORBA naming service groups

Note: The two special groups are already defined and have roles assigned. To change the roles assigned to the two special groups, click the group name (link).

2. To add a new group, click **Add**.


Group: [✓ Specify group:] Administrators

Select from special subject: EVERYONE

Role(s): Cos Naming Read, Cos Naming Write, Cos Naming Create, [✓ Cos Naming Delete]

Figure 10-14 Assign an authorization level

3. Select the **Specify Group** button, enter a group name (case-sensitive) and select an authorization level (role). The group must be a valid user in the active user registry. If you have not activated global security, the local operating system user registry will be used. Remember, before these settings take effect, you will have to enable and configure WebSphere global security.

   To specify multiple roles, hold the Ctrl key while you click the applicable roles.

4. Click **Apply**.
5. Click **OK**.

11

Asynchronous messaging

In this chapter, we describe the concepts behind the asynchronous messaging functionality provided as part of WebSphere Application Server V5. We cover:

- Asynchronous messaging
- WebSphere support for asynchronous messaging
- WebSphere JMS provider (embedded)
- WebSphere MQ JMS provider
- Generic JMS provider
- WebSphere clusters and MQ clusters
- JMS scenarios
- Configuration of JMS resources

11.1 Asynchronous messaging

Asynchronous messaging provides a method for communication based on the Java Message Service (JMS) programming interface. JMS provides a common mechanism for Java programs (clients and J2EE applications) to create, send, receive, and read asynchronous requests, as JMS messages.

11.1.1 Concepts

The base support for asynchronous messaging using JMS includes the common set of JMS interfaces and associated semantics that define how a JMS client can access the facilities of a JMS provider. This support enables J2EE applications, as JMS clients, to exchange messages asynchronously with other JMS clients by using JMS destinations (queues or topics).

A J2EE application can use JMS queue destinations for point-to-point messaging and JMS topic destinations for publish/subscribe messaging.

A J2EE application can explicitly poll for messages on a destination then retrieve messages for processing by business logic beans.The business logic beans use standard JMS calls to process the message, for example to extract data or to send the message on to another JMS destination.

With the base JMS/XA support, the J2EE application uses standard JMS calls to process messages, including any responses or outbound messaging. Responses can be handled by an enterprise bean acting as a sender bean, or handled in the enterprise bean that receives the incoming messages. Optionally, this process can use two-phase commit within the scope of a transaction. This level of functionality for asynchronous messaging is called bean-managed messaging, and gives an enterprise bean complete control over the messaging infrastructure, for example for connection and session pool management. The common container has no role in bean-managed messaging.

Message-driven beans

Application server products compliant with the J2EE 1.3 specification provide support for automatic asynchronous messaging using message-driven beans (MDBs), a type of enterprise bean defined in the EJB 2.0 specification. Messages are automatically retrieved from JMS destinations, optionally within a transaction, then sent to the MDB in a J2EE application, without the application having to explicitly poll JMS destinations.

Extended messaging

J2EE 1.3 compliant application server products may optionally provide J2EE applications with another level of functionality for asynchronous messaging called *extended messaging*. The common container manages the messaging infrastructure, and extra standard types of messaging beans are provided to add functionality to that provided by MDBs. This level of functionality enables application developers to concentrate on the business logic to be implemented by the enterprise beans and to leave the messaging usage to standard messaging objects and configuration of the common container.

Extended messaging is provided as a programming model enhancement in IBM WebSphere Application Server Enterprise V5.0 and in WebSphere Business Integration Server Foundation V5.1.

11.1.2 Application usage

Applications can use the following styles of asynchronous messaging:

- Point-to-point

  Point-to-point applications use queues to pass messages between each other. The applications are called point-to-point, because a client sends a message to a specific queue and the message is picked up and processed by a server listening to that queue. It is common for a client to have all its messages delivered to one queue. A queue can contain a mixture of messages of different types.

- Publish/subscribe

  Publish/subscribe systems provide named collection points for messages, called topics. To send messages, applications publish messages to topics. To receive messages, applications subscribe to topics; when a message is published to a topic, it is automatically sent to all the applications that are subscribers of that topic. By using a topic as an intermediary, message publishers are kept independent of subscribers.

Both styles of messaging can be used in the same application.

Applications can use asynchronous messaging in the following ways:

- One-way

  An application sends a message and does not require a response. This pattern of use is often referred to as a datagram.

- Request/response

  An application sends a request to another application and expects to receive a response in return.

- One-way and forward

  An application sends a request to another application, which sends a message to yet another application.

These messaging techniques can be combined to produce a variety of asynchronous messaging scenarios.

11.1.3 Java Message Service (JMS)

JMS supports the development of message-based applications in the Java programming language, allowing for the asynchronous exchange of data and events throughout an enterprise.

JMS supports both the point-to-point and publish/subscribe messaging models.

Note: An application server product compliant with the J2EE 1.3 specification is required to provide support for both the JMS API and a built-in JMS provider implementation. WebSphere V4 only provides support for the JMS API. A JMS provider implementation has to be installed separately.

Components

The main components of WebSphere Application Server's implementation of JMS are shown in Figure 11-1 on page 454, from the JMS provider, through a connection to a destination, then to an application (acting as a JMS client) that processes the message retrieved from the destination.

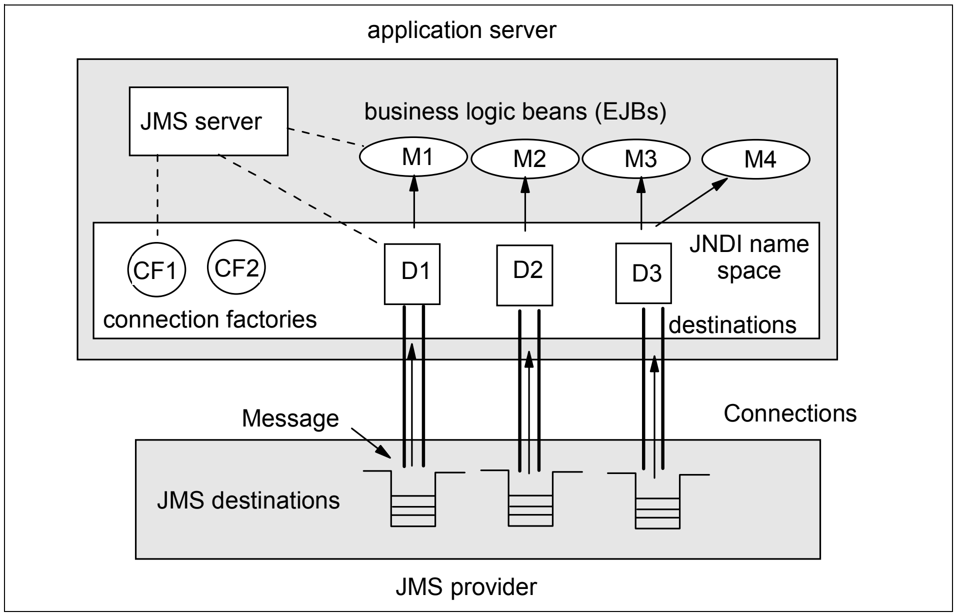


Figure 11-1 JMS components

- **JMS provider**: A base messaging system and related Java classes that implement the JMS API. The resources of the JMS provider are accessed through JMS connection factory and destination objects.
- **JMS server**: The JMS functions of the JMS provider are accessed by the JMS server within the application server. Each JMS server is responsible for managing a queue manager and a broker. The queue manager is the provider of the point-to-point (queue) service. The broker is the provider of the publish/subscribe (topic) service.
- **JMS administered objects**: In JMS, certain objects are created by the administrator and stored in a directory. The Java Naming and Directory Interface (JNDI) name space is used to hold references to JMS administered objects, encapsulating settings necessary for connecting to and using the queues and/or topics of messaging systems. The JMS administered objects are:
  - **JMS connection factories**: A connection factory is used to create connections with the JMS provider for a specific JMS queue or topic destination. Each connection factory encapsulates the configuration parameters needed to create a connection to a JMS destination. Storing the connection details in JNDI makes the application connection code vendor independent. There are two types of JMS connection factory:
    - QueueConnectionFactory: Encapsulates the settings necessary to connect to a queue-based messaging system.
    - TopicConnectionFactory: Encapsulates the settings necessary to connect to a topic-based messaging system.
  - **JMS destinations:** A JMS destination provides a specific endpoint for messages. A J2EE application can explicitly poll for messages on a destination then retrieve messages for processing. Destination objects are references to JMS queues and topics. Storing these objects in JNDI separates the application from knowledge of the queueing topology or topic space.

Using J2EE references with JMS

The J2EE platform adds a further level of indirection to JMS administered objects:

- Alias references are used by the application code.
- Aliases are mapped to real JMS administered objects during application deployment.

It is possible to deploy multiple independently developed applications into the same application server, with each application using the same names for JMS queues or topics.

To avoid name collision, J2EE uses the *java:comp/env* mechanism. In this mechanism, each application looks up the name in its own personal (java:comp/env) environment. The administrator can define different resolutions of the lookup name for each application, and makes the container responsible for returning the correct references from JNDI for each local lookup.

Example 11-1 shows an example of accessing the JMS queue connection factory and destination using the J2EE local application (java:comp/env) JNDI name space.

Example 11-1 Using JMS with java:comp/env

```
import javax.jms.*;
import javax.naming.*

// Get the JNDI initial context
InitialContext initCtx = new InitialContext();

// get the queue connection factory
QueueConnectionFactory qcf = (QueueConnectionFactory)
initCtx.lookup("java:comp/env/jms/myQCF");

// Create a connection
QueueConnection conn = qcf.createQueueConnection();

// Create a JMS session
QueueSession session = conn.createQueueSession(false,
Session.AUTO_ACKNOWLEDGE);

// get the queue used to send a message
Queue q = (Queue) lookup("java:comp/env/jms/myQueue");

// Send a message...
QueueSender sender = session.createSender(q);
sender.send(session.createTextMessage("Hello world"));
sender.close();
```

In the above example, the local resource references jms/myQCF and jms/myQueue would be configured on application packaging or deployment to refer to QueueConnectionFactory and Queue JMS administered objects already configured into the WebSphere Application Server name space.

11.1.4 Message-driven beans

Note: WebSphere V4 does not provide support for MDBs, although the Enterprise Edition provides a “work-alike” called message beans that leverages stateless session EJBs. However, the WebSphere V4 container does not have these services integrated and does not implement the EJB 2.0 specification.

The support for MDBs is based on the message listener, which comprises the following components:

- **Listener manager**

  Controls and monitors one or more listeners.

- **Listener**

  Each listener monitors a JMS destination for incoming messages. When a message arrives on the destination, the listener passes the message to a new instance of a user-developed MDB for processing. The listener then looks for the next message without waiting for the MDB to return.

- **JMS destination**

  Represents a queue or topic.

- **Message-driven beans**

  A stateless component that is invoked by the J2EE container as a result of the arrival of a JMS message at a particular JMS destination (queue or topic). The MDB is in a sense triggered by the arrival of the message.

  Messages arriving at a destination being processed by a listener have no client credentials associated with them; the messages are anonymous. To enable some level of security, a listener assumes the credentials of the application server process for the invocation of the MDB.

  It is recommended that MDBs be developed so that they delegate the business processing of incoming messages to another enterprise bean (a business logic bean), to provide clear separation of message handling and business processing. This also enables the business logic to be invoked by either the arrival of incoming messages or by direct invocation of the EJB. For example, the business logic bean could also be invoked directly by EJB clients such as: Web container based Java classes and servlets, J2EE application clients.

The components of the recommended MDB architecture are shown in Figure 11-2 on page 458.

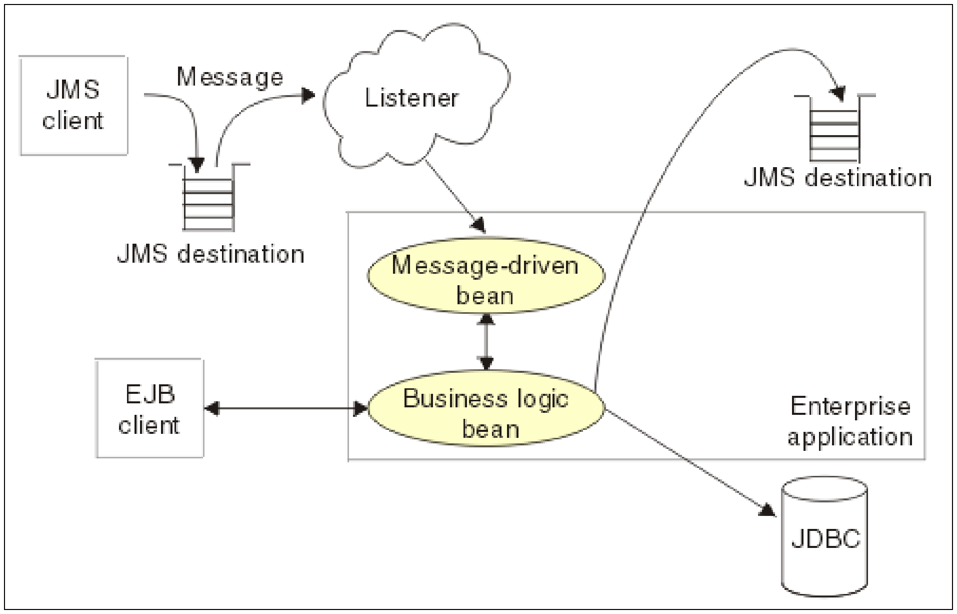


Figure 11-2 Messaging using message-driven beans

MDBs can handle messages read from JMS destinations within the scope of a transaction. If transaction handling is specified for a JMS destination, the JMS listener starts a global transaction before it reads any incoming message from that destination. When the MDB processing has finished, the JMS listener commits or rolls back the transaction (using JTA transaction control).

11.2 WebSphere support for asynchronous messaging

WebSphere Application Server V5 support for asynchronous messaging can be grouped into the following categories:

- Support for J2EE 1.3.
- Support for Java Message Service (JMS).
- Support for message-driven beans (MDBs).

11.2.1 WebSphere support for J2EE 1.3

The changes relating to asynchronous messaging in WebSphere Application Server V5 are brought about by the requirements of the J2EE 1.3 specification.

In order to be compliant with the specification, an application server product must:

1. Provide a JMS provider implementation.

   In J2EE 1.3, the JMS API becomes an integral part of the J2EE platform.

   WebSphere includes an embedded JMS server that can be chosen as an option for the base installation.

2. Support the message-driven beans programming model, defined in the EJB 2.0 specification.

   WebSphere includes an implementation of MDBs that is fully compliant with the EJB 2.0 specification.

11.2.2 WebSphere support for JMS

WebSphere provides support for JMS in the following areas.

JMS providers

The JMS functionality provided by WebSphere includes support for three types of JMS provider:

- WebSphere JMS provider (embedded messaging)

  In WebSphere Application Server V4, JMS support required the installation of a separate JMS product, for example WebSphere MQ for point-to-point messaging and either WebSphere MQ Integrator or the MA0C product extension for publish/subscribe messaging.

  WebSphere Application Server V5 includes an embedded WebSphere JMS provider that is integrated with the product and can be chosen as an installation option.

  The implementation of the embedded WebSphere JMS provider is based on the integration of IBM's WebSphere MQ product, the WebSphere MQ Event Broker product and the JMS Client ("MA88") support pack into WebSphere Application Server. The versions of these products that are integrated with WebSphere Application Server are a reduced footprint of the independently shipped MQ products.

- WebSphere MQ JMS provider

  WebSphere Application Server V5 supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

- Generic JMS providers

  WebSphere Application Server V5 supports the use of generic JMS providers, as long as they implement the ASF component of the JMS 1.0.2 specification.

Note:

1. There can be more than one JMS provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of the WebSphere JMS provider, WebSphere MQ JMS provider and a generic JMS provider.
2. It is possible to have both the internal JMS provider and full WebSphere MQ concurrently installed on the same machine. However, there will only be one copy of the JMS client classes (MA88) and MQ transport code (binaries). This has the consequence that both must be at the same version and patch level.
3. There can only be one JMS server process (embedded JMS provider) per node.

The WebSphere administration clients provide configuration and/or management of JMS resources for each of these types of providers:

Table 11-1 JMS providers versus JMS resources

JMS provider	**Provider resource**	**Configurable JMS resources**
Integral	WebSphere JMS provider	WebSphere Queue Connection Factories WebSphere Topic Connection Factories WebSphere Queue Destinations WebSphere Topic Destinations
WebSphere MQ	WebSphere MQ JMS provider	WebSphere MQ Queue Connection Factories WebSphere MQ Topic Connection Factories WebSphere MQ Queue Destinations WebSphere MQ Topic Destinations
Generic	Generic JMS provider	Not configurable - can only set up InitialContextFactory to access externally configured resources.

JMS administered objects

The tool used to add the provider's administered objects to a JNDI name space is dependent upon the name space targeted:

- WebSphere name space

  Generic JMS providers would need to provide an administration tool that can access the WebSphere name space. The JMS objects of the embedded WebSphere JMS provider and WebSphere MQ JMS providers can be administered directly using the WebSphere administration tools.

- External name space

  Embedded and WebSphere MQ resources need to be registered with the WebSphere name space. Generic JMS providers can register their JMS administered objects in external name spaces as long as their tool has support for that name space.

JNDI name space

The JNDI name space can be either:

- The federated name space of WebSphere.
- An external JNDI name space.

In order for an application hosted by WebSphere to access the JMS administered objects from the JNDI name space, the JMS provider resource must be configured in WebSphere. The JMS provider resource encapsulates the information, shown in Table 11-2, necessary for making connection to the context root of the appropriate JNDI name space.

Table 11-2 JMS provider settings

Setting	WebSphere JMS provider	WebSphere MQ JMS provider	Generic JMS provider
External Initial Context Factory	Not configurable	Not configurable	Configurable
External Provider URL	Not configurable	Not configurable	Configurable
Classpath	Not configurable	Not configurable	Configurable
Native library path	Not configurable	Not configurable	Configurable

As shown in Figure 11-3, the WebSphere V5 support for generic JMS providers works by registered "alias" entries, in the application server's local name space, which refer to the real JMS administered objects (JMS connection factory and JMS destination) in the external name space.

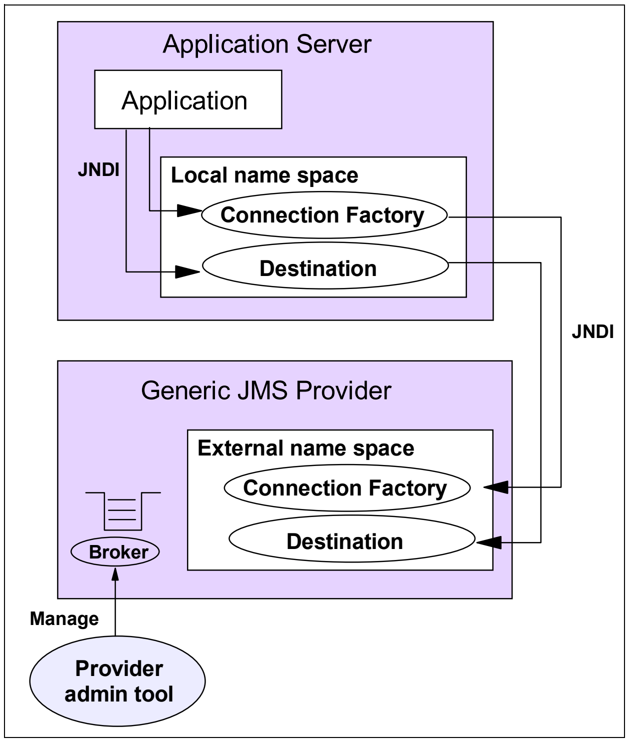


Figure 11-3 Generic JMS provider components

JMS transaction integration

In order for JMS operations to be part of a global transaction, the container must be made aware of the JMS session used by the operations. WebSphere V5 uses the J2C Connection Manager runtime to provide the container services necessary to support transactions.

Note: WebSphere V4 implements a solution to this problem specifically for WebSphere MQ by wrappering the MQ ConnectionFactory, Connection and Session objects with WebSphere-specific versions stored in JNDI in place of the formal WebSphere MQ objects. However, if the non-WebSphere specific JMS classes were used, then any transactions in an application's JMS code could not be handled by the WebSphere V4 container, preventing the application from participating in global (two-phase commit) transactions.

JMS resource pooling

In WebSphere V5, the JMS connection and session resources are managed by the J2C Connection Manager. The advantage of this approach is that it enables the integration of container services such as transaction and security.

JMS administration tools

The support provided by WebSphere administration tools for configuration of JMS providers differs depending upon the provider. Table 11-3 provides a summary of the support.

Table 11-3 WebSphere administration support for JMS provider configuration

Configurable objects	WebSphere JMS provider	WebSphere MQ JMS provider	Generic JMS provider
Initial context factory and provider URL	N **	N **	Y
Messaging system objects (queues/topics)	Y	N	N
JMS administered objects (JMS connection factory and JMS destination)	Y	Y	N
** The settings are not exposed in the WebSphere administrative console.			

The key points are:

- WebSphere provides functionality so that JMS administered objects for both the WebSphere JMS provider and WebSphere MQ JMS provider can be configured using the WebSphere administration tools. There is no need to use the JMSAdmin tool, provided with the WebSphere MQ MA88 JMS extension, to administer WebSphere MQ JMS provider resources.
- The WebSphere administration tools can be used to configure and manage the actual queue manager, queues and topics of the internal JMS provider only.
- For the WebSphere MQ and generic JMS providers, the messaging system and its resources must be configured using the provider’s native tools. The WebSphere administration tools do not provide this functionality.

11.2.3 WebSphere support for message-driven beans

WebSphere provides support for MDBs in the following areas.

MDB components

WebSphere V5 support for MDBs is based on JMS message listeners and the message listener service. The main components are:

- **Message listener service**

  An extension to the JMS functions of the JMS provider provides a listener manager, which controls and monitors one or more JMS listeners.

- **Listener**

  Each listener monitors either a JMS queue destination (for point-to-point messaging) or a JMS topic destination (for publish/subscribe messaging). The listener is the part of the application server that invokes the MDB when the message arrives. It is fully integrated with the EJB container.

  Listener recovery from messaging domain failures is new to WebSphere Application Server V5. For example, if the external queue manager fails in WebSphere Application Server V4, the listener stops and must be manually restarted. In WebSphere Application Server V5, the listener automatically recovers in such scenarios.

  Each listener completes several steps for the JMS destination that it is to monitor, including:

  – Creating a JMS server session pool, and allocating JMS server sessions and session threads for incoming messages.

  – Interfacing with JMS ASF to create JMS connection consumers to listen for incoming messages.

  – If specified, starting a transaction and requesting that it is committed (or rolled back) when the EJB method has completed.

  – Processing incoming messages by invoking the onMessage() method of the specified MDB.

- **Connection factory**

  Used to create connections with the JMS provider. Each connection factory encapsulates the configuration parameters needed to create a connection to a JMS provider.

- **Listener port**

  Defines the association between a connection factory, a destination, and a deployed MDB. Listener ports are used to simplify the administration of the associations between these resources.

Note: Listener ports are a WebSphere V5 specific component. They are not defined as part of the J2EE 1.3 specification. Consequently, application servers other than WebSphere V5 may provide very different mechanisms for associating an MDB with a connection factory and a JMS destination.

Figure 11-4 summarizes the WebSphere V5 MDB components.

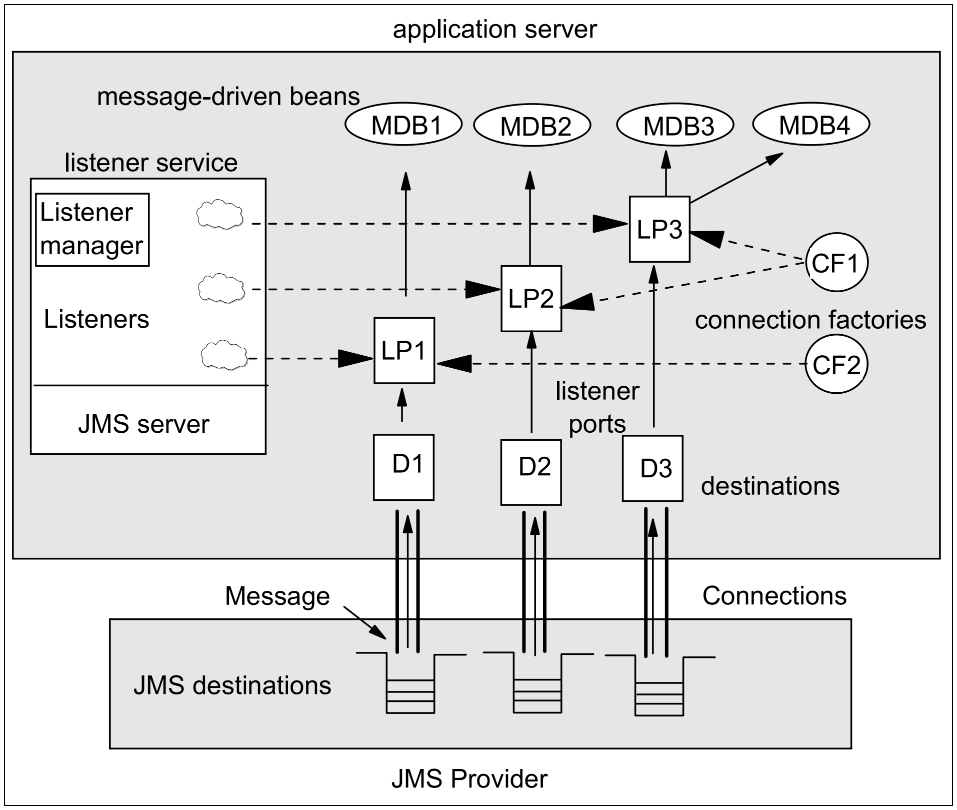


Figure 11-4 Components of WebSphere MDB implementation

MDB resource configuration

The configuration of MDBs is fully integrated into the WebSphere administration tools.

When a deployed MDB is installed, it is associated with a listener port and the listener for a destination. When a message arrives on the destination, the listener passes the message to a new instance of a MDB for processing.

When an application server is started, it initializes the listener manager based on the configuration data. The listener manager creates a dynamic session thread pool for use by listeners, creates and starts listeners, and, during server termination, controls the cleanup of listener message service resources.

Migration of WebSphere V4 message beans

WebSphere Application Server V5 provides the mb2mdb script for use in migrating WebSphere V4 message beans to fully compliant MDBs. In addition, the samples provided with WebSphere have been migrated to use MDBs.

For details on the mb2mdb script and its use, see the WebSphere Application Server V5 InfoCenter and the redbook *Migrating to WebSphere V5.0: An End-to-End Migration Guide,* SG24-6910.

11.3 WebSphere JMS provider

WebSphere includes an embedded JMS provider called the WebSphere JMS provider. This provider is composed of the following IBM products:

- WebSphere MQ
- WebSphere MQ MA88 SupportPac - JMS client classes
- An early release of WebSphere Event Broker, providing the publish/subscribe service

Each of these is reduced in footprint and function compared to the independently available product.

11.3.1 Recommended use

The WebSphere JMS provider is recommended for use in the following scenarios:

- When the functionality of full WebSphere MQ is not required

  This will simplify configuration and management of the messaging functionality.

- When first introducing messaging into applications

  The choice of using the WebSphere JMS provider does not prevent the full WebSphere MQ being installed, configured, and used as the JMS provider at a later date as operators become more familiar with messaging. Reinstallation of WebSphere is not required; see the WebSphere Application Server InfoCenter section “Moving from the embedded WebSphere JMS provider to WebSphere MQ” for more information on this topic.

Note: The use of the WebSphere JMS provider does not change existing application usage scenarios for applications using JMS.

11.3.2 Key features

There are several features that are common to both the point-to-point and publish/subscribe functions:

- The WebSphere JMS provider supports the requirements of the J2EE 1.3 specification:
  - Support for the JMS 1.0.2 specification.
  - Full compliance with the J2EE 1.3 compliance tests.
- The WebSphere JMS provider does not support:
  - APIs other than those defined in the JMS API.
  - Interoperability with WebSphere MQ queue managers.
- Security is integrated with the WebSphere security system, with user and group access being configured using the WebSphere administration tools.
- The JMS server hosts the broker and manages the external WebSphere MQ processes, including creation, management and deletion of the WebSphere MQ queues and topics.
- All JMS client access, either point-to-point or publish/subscribe, is performed using WebSphere MQ client (TCP/IP) mode.

  Note: This is a consequence of the use of a SVRCONN channel for all client access. The SVRCONN channel allows a security exit to be configured to link the WebSphere MQ native process into the WebSphere security system. The JMS server provides a security listener port (when WebSphere global security is enabled) that is called by the security exit to pass security requests through to WebSphere.

- Tracing is integrated with the WebSphere tracing infrastructure, with tracing configuration performed via the WebSphere administration tools.

Point-to-point messaging

The point-to-point functionality of the WebSphere JMS provider is provided by a built-in WebSphere MQ server, providing a WebSphere MQ queue manager and queues. All point-to-point JMS clients access the queue manager listener port directly, using WebSphere MQ client (TCP/IP) mode. The JMS server is not part of the messaging communication.

Publish/subscribe messaging

The publish/subscribe functionality of the WebSphere JMS provider includes two different access mechanisms:

- **Direct:** Used for direct access to the broker, bypassing any queueing mechanisms in order to provide high performance. It does not provide full-function publish/subscribe, providing only non-persistent, non-durable, non-transactional subscriptions.
- **Queued:** Used for full-function publish/subscribe, providing support for durable, persistent, transactional subscriptions. This mechanism accesses the broker via the queue manager, which is configured to support the broker.

11.3.3 Limitations

The major limitations of the WebSphere JMS provider are:

- It is not possible to exchange messages with queue managers outside WebSphere.

  Note: This limitation does not preclude remote JMS clients from accessing the internal JMS provider of a WebSphere V5 node. It only prevents the queue manager of the internal JMS provider from exchanging messages with other queue managers, either within the same cell (administrative domain) or outside the cell.

- Limited publish/subscribe broker capabilities, including:
  - No message flows.
  - No message transformation.
  - No database support.
  - No shared topic spaces.
- Queue manager administration and configuration are not provided by the WebSphere administration tools. Administration of the queue manager and channels requires the use of the createmq and deletemq scripts provided with the WebSphere installation.

  Note: It is not possible or required for the user to directly administer (create, update or delete) the WebSphere MQ queues and topics. The JMS server performs these functions.

- It is pre-installed onto each node, but none of its attributes are configurable using the WebSphere administration tools.
- No mechanism is provided for sending test messages to queues.

- No mechanism is provided for clearing messages from queues.
- Queue manager clustering is not supported.
- No support for JMS server failover.

11.3.4 Runtime components

The embedded WebSphere JMS provider is fully integrated with the WebSphere runtime. The provider runs in one of two modes (base or Network Deployment), depending upon whether the node has been added to a Network Deployment cell (administrative domain).

Base configuration

Figure 11-5 on page 469 shows the internal messaging components present in a base WebSphere configuration, that is, an application server that has not been added to a Network Deployment cell.

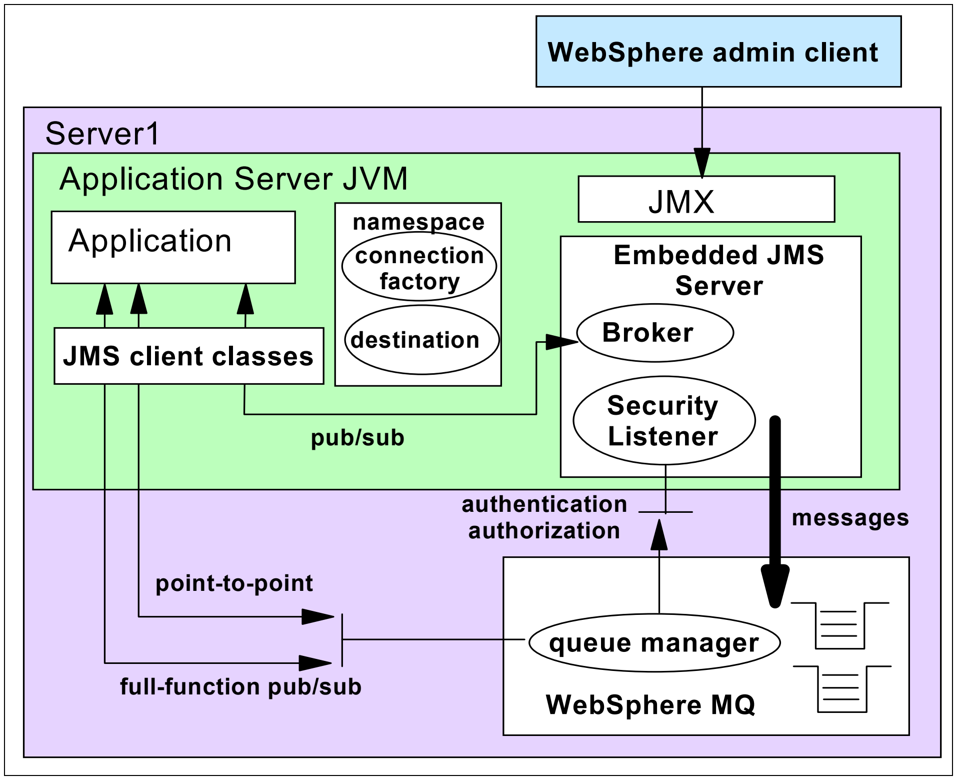


Figure 11-5 Base configuration internal JMS server components

The key points of the base configuration are:

- The JMS server runs embedded in the application server JVM, with the broker hosted within the JVM and the queue-based messaging system running as native WebSphere MQ processes, separate from the JVM.
- The JMS server manages the external WebSphere MQ processes, providing management functions to:
  - Start and stop the queue manager.
  - Manage the queue manager, for example define new queues.
  - Provide a security listener port that the WebSphere MQ security exit uses to connect to the WebSphere security system. This is used to provide authentication and authorization checks when WebSphere security is enabled.
- The security port is defined as part of the JMS server component of the application server (server1).

  Note: Since the WebSphere MQ processes are native, running outside the WebSphere application server JVM, they do not have access to the WebSphere security system. The embedded messaging component provides the listener in order to forward security checks to WebSphere on behalf of WebSphere MQ.

- Point-to-point messaging involves a client using the JMS client classes to make a client mode (TCP/IP) request directly to the local internal WebSphere MQ queue manager. The internal JMS provider does not use bind mode, even though the WebSphere MQ server is on the same machine (local).

  The port number of the queue manager listener port is defined by the JMSSERVER_QUEUED_ADDRESS value defined within the application server’s entry in the serverindex.xml configuration file.
- The default publish/subscribe messaging involves a client using the JMS client classes (MA88) to make a direct call on the broker running within the application server JVM. The broker listens on a TCP/IP port for incoming direct requests.

  The port number of the broker direct port is defined by the JMSSERVER_DIRECT_ADDRESS value defined within the application server’s entry in the serverindex.xml configuration file.
- Full-function publish/subscribe messaging involves a client using the JMS client classes (MA88) to make a client mode (TCP/IP) publishing request to the queue manager listener port, rather than directly accessing the broker.

The internal queue manager is configured to use the broker to handle any publish requests it receives.

- All administration and management of the embedded JMS server, broker ,and WebSphere MQ is performed via the WebSphere administration tools.

  The WebSphere administration tools access the embedded JMS server by connecting to the JMX management port(s) of the application server, defined by SOAP_CONNECTOR_ADDRESS. This port is defined within the application server's entry in the serverindex.xml configuration file.

- All JMS administered objects are registered in the application server's local JNDI name space, using the WebSphere administration tools.

  The definitions of the JMS administered objects are stored in the resources definition file (resources.xml) at the scope (cell, node or managed server) at which it was created.

  Note:

  - If a JMS resource is defined at the cell scope, it will be available to each application server managed in that cell. Each application server will separately configure that resource and register it in its local application server name space.
  - If a JMS resource is defined at the node scope, it will be available to each application server managed by that node. Each application server will separately configure that resource and register it in its local application server name space.

- All WebSphere MQ queues are created, updated, or deleted using the WebSphere administration tools. The standard WebSphere MQ tools are not available for this task.

Network Deployment configuration

Figure 11-6 on page 472 shows the internal messaging components present in a Network Deployment node configuration, that is, an application server that has been added to a Network Deployment cell.

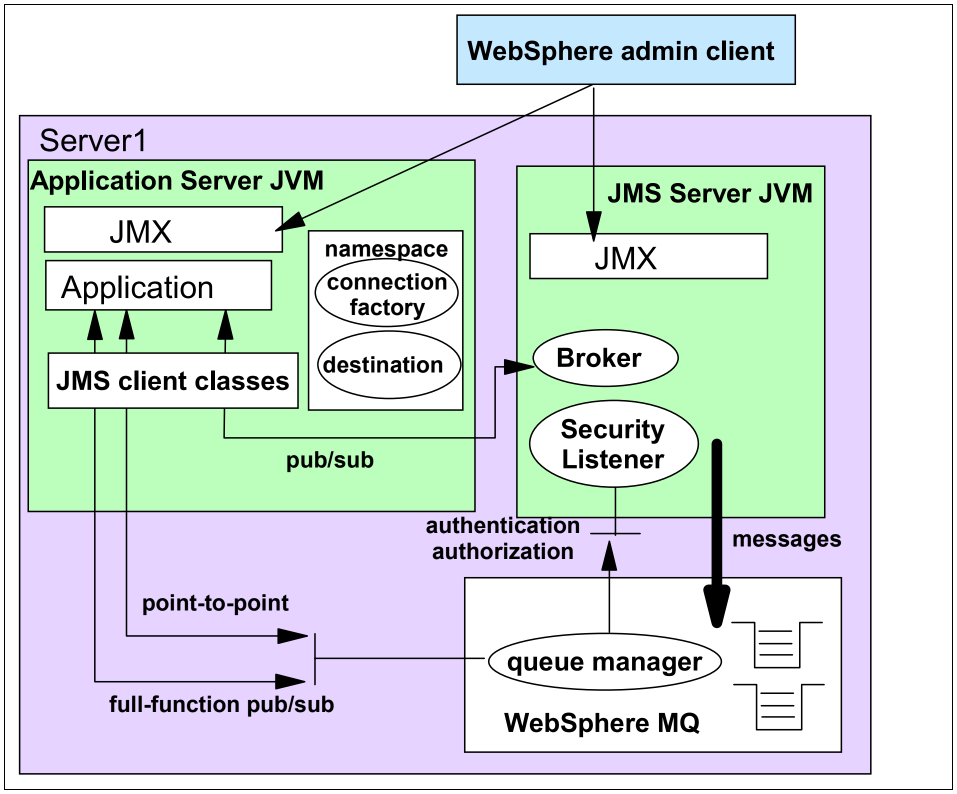


Figure 11-6 Network Deployment configuration of JMS server components

The Network Deployment configuration differs from the base configuration in the following areas:

- The JMS server runs in its own dedicated JVM (jmsserver) that is created as part of adding a node to a cell. The jmsserver is a WebSphere Application Server V5 managed process, having its own server definition (server.xml) in the node's configuration directory, as well as dedicated IP ports defined in the node's serverindex.xml configuration file.

  See Chapter 7, "Administration overview" on page 205 for details.

- The jmsserver managed process is a specialized server instance, since its configuration does not include:
  - A Web container
  - An EJB container
  - A name service
  - An Object Request Broker (ORB)

Notes:

1. There can only be one JMS server (WebSphere JMS provider) per node.
2. There can be more than one JMS provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of WebSphere JMS provider, WebSphere MQ JMS provider, and generic JMS provider.

- The JMS server on a node is independent of the JMS servers on all other nodes in a cell.
- The dedicated jmsserver process manages the external WebSphere MQ processes, providing management functions to:
  - Start and stop the queue manager.
  - Manage the queue manager, for example, to define new queues and topics.
- The IP ports used for point-to-point, direct publish/subscribe, and queued publish/subscribe messaging are defined with the jmsserver entry in the serverindex.xml configuration file, rather than an application server entry as occurs with the base configuration.
- All JMS administered objects are registered, using the WebSphere administration tools, in the local JNDI name space of each application server that hosts an application that uses the messaging functionality of the JMS server process.
- JMS resources (queues and topics) defined in a particular node's JMS server are accessible from anywhere in the cell, rather than just the local application server.

11.3.5 Administration

The WebSphere JMS provider is fully integrated with the WebSphere administration. In fact, the administration tools that come standard with WebSphere MQ are not provided with the WebSphere MQ included with the WebSphere JMS provider.

Note: The JMS server is installed using default WebSphere MQ settings which may not meet high load requirements. However, manual editing of the MQ configuration files can be performed to adjust MQ settings that are not exposed by WebSphere administration tools. See Chapter 11 of *WebSphere MQ System Administration Guide*, SC33-1873-02 for details.

11.3.6 Security considerations

Security for the WebSphere JMS provider operates as a part of WebSphere global security, and is enabled only when global security is enabled.

Enabling global security has the following impacts on the use of the WebSphere JMS provider:

- All JMS connections are authenticated. If authentication is successful, then the JMS connection is created; if authentication fails, then the connection request is ended.
- Access to JMS resources owned by the WebSphere JMS provider is controlled via access authorizations.

Authentication

As mentioned in 11.2.2, “WebSphere support for JMS” on page 459, the JMS server uses the J2C Connection Manager runtime for underlying services. As a result, JMS connection authentication uses J2C authentication. This involves:

- Configuration of a J2C authentication alias using the WebSphere administration tools. The alias encapsulates the user ID and password required for access to the resource.

  > **Note:** User IDs longer than 12 characters are not supported for authentication of WebSphere JMS provider resources. J2C authentication aliases must be configured to use user IDs of 12 characters or less.

- Configuration of the JMS connection factory resource, using the WebSphere administration tools, to use the J2C authentication alias.

  Both a component-managed authentication alias and a container-managed authentication alias setting can be configured for each connection factory.

  The specific setting to use depends upon the authentication type used on the connection factory resource reference configured in the application’s deployment descriptor. The possible authentication types that can be used for the <resource-ref> element’s <res-auth> values are:

  – **Application**

    Use the component-managed authentication alias setting. Use the WebSphere administration tools to assign the new J2C authentication alias as the value of this setting.

    If the application that tries to create a JMS connection specifies a user ID and password, those values will be used to authenticate the connection creation request.

If the application does not specify a user ID and password, the values defined by the component-managed authentication alias will be used.

If the connection factory is not configured with a component-managed authentication alias, then a runtime JMS exception will be returned when the application attempts to open a JMS connection.

– **Container**

Use the container-managed authentication alias setting. Use the WebSphere administration tools to assign the new J2C authentication alias as the value of this setting.

The user ID and password values defined by the container-managed authentication alias will be used to authenticate the JMS connection creation request. The application does not provide the user name and password.

If a container-managed authentication alias is not defined, then a runtime JMS exception will be returned when the application attempts to open a JMS connection.

Authorization

Authorization to access JMS resources owned by the WebSphere JMS provider is controlled by settings stored in the integral-jms-authorizations.xml file located under <WAS_HOME>/config/cells/<cellname>. The settings are summarized in Table 11-4.

Note: The authorization settings are not configurable nor viewable via the WebSphere administration tools. The only mechanism currently provided is to manually edit the appropriate authorizations file.

Table 11-4 Authorization rights for JMS destinations

Object	Authorization rights
Queue	Read Write
Topic	Publish Subscribe Persist

For cell-wide authorizations, edit the file in the cell directory:

<WAS_HOME>/config/cells/<cellname>/integral-jms-authorizations.xml

For node-wide authorizations, copy the cell-wide file into the node directory and edit its settings as required:

<WAS_HOME>/config/cells/<cellname>/nodes/<nodename>/integral-jms-authorizations.xml

An example is shown in Example 11-2.

Example 11-2 integral-jms-authorizations.xml file

```
<integral-jms-authorizations>
  <dynamic-update>true</dynamic-update>

  <queue-admin-userids>
    <userid>adminid1</userid>
    <userid>adminid2</userid>
  </queue-admin-userids>

  <queue-default-permissions>
    <permission>write</permission>
  </queue-default-permissions>

  <queue>
    <name>q1</name>
    <public>
    </public>
    <authorize>
      <userid>useridr</userid>
      <permission>read</permission>
    </authorize>
    <authorize>
      <userid>useridw</userid>
      <permission>write</permission>
    </authorize>
  </queue>

  <topic>
    <name>a/b/c</name>
    <public>
      <permission>+sub</permission>
    </public>
    <authorize>
      <userid>useridpub</userid>
      <permission>+pub</permission>
    </authorize>
  </topic>
</integral-jms-authorizations>
```

For further details on the use and content of the integral-jms-authorizations.xml file, see the documentation comments included at the top of the file and the *IBM WebSphere V5.0 Security Handbook,* SG24-6573.

11.3.7 Interoperability

The WebSphere JMS provider does not interoperate with other WebSphere MQ queue managers. As such, it cannot be used for heterogeneous WebSphere MQ environments.

The WebSphere JMS provider is able to interoperate with the following clients:

1. JSPs, servlets and EJBs running in WebSphere Application Server V5

   Any application server-based client can directly use any JMS server in the cell, simply by using the JMS connection factory and JMS destination associated with the chosen JMS server.

2. WebSphere Application Server V5 J2EE application clients

   Any J2EE application client can use any JMS server in the cell, simply by:

   – Using the JMS connection factory and JMS destination associated with the chosen JMS server.

   – Using the JMS client classes (MA88) delivered with WebSphere V5.

3. WebSphere Application Server V4 clients

   A client hosted in a WebSphere Application Server V4 environment can make use of the WebSphere JMS provider, as long as:

   – WebSphere Application Server V4 is at the correct patch level for JNDI interoperability with WebSphere Application Server V5.

   – WebSphere Application Server V4 has been upgraded to use a version of the JMS client classes compatible with WebSphere Application Server V5.

   – The JMS administered objects have configured bindings (aliases) created by the administrator in the WebSphere Application Server V5 legacy name space root.

   See Chapter 10, “WebSphere naming implementation” on page 399 for further details on WebSphere Application Server V4 client interoperability with WebSphere Application Server V5.

11.4 WebSphere MQ JMS provider

In order to use the WebSphere MQ JMS provider as an external JMS provider with WebSphere, the following components must be installed, in addition to the WebSphere installation:

- WebSphere MQ
- WebSphere MQ MA88 SupportPac - JMS classes.
- Broker for publish/subscribe messaging. There are three alternatives that will work with WebSphere:
  - WebSphere MQ Publish and Subscribe
  - WebSphere MQ Integrator
  - WebSphere MQ MA0C SupportPac - Publish/Subscribe

Notes:

1. The version of each of these products that is integrated into the WebSphere JMS provider is reduced in footprint and function compared to the independently available product.
2. The use of MA0C is discouraged, since the other brokers provide a much more robust production publish/subscribe environment.

Although WebSphere provides an embedded WebSphere JMS provider based on WebSphere MQ, it is expected that in most instances the WebSphere MQ JMS provider would be the preferred choice. Reasons for this are:

- The WebSphere JMS provider can only be used from within a WebSphere environment. However, many production environments use WebSphere MQ for heterogeneous integration, allowing J2EE applications to communicate with other MQ applications that may be written in languages other than Java and/or hosted in environments other than WebSphere.
- WebSphere MQ supports advanced messaging topologies, including:
  - Queue manager clustering

    Supports higher throughput levels and continuous availability for new messages, for example those messages that trigger MDBs.
  - High availability technologies

    Use shared disks to enable both the messaging service and individual in-flight messages to be highly available. Such technologies include HACMP on AIX, Microsoft Cluster Server on Windows, and SunCluster on Solaris.

  - Broker collectives

    This uses multiple copies of WebSphere MQ Integrator.
  - z/OS shared queues

    High availability messaging.
- Customers currently using full WebSphere MQ will be more familiar with its issues and management. There is no need to become familiar with a custom form of WebSphere MQ.

11.4.1 Key features

The WebSphere MQ JMS provider supplies the following features that are not supported by the WebSphere JMS provider of WebSphere Application Server V5:

- Full support for the WebSphere MQ API.
- Support for communication with other WebSphere MQ applications, not just those applications hosted in a WebSphere environment. This includes communication with JMS and non-JMS applications.
- Support for WebSphere MQ queue manager and channels.
- Support for WebSphere MQ clusters.
- Multi-broker capability.
- Support for integration with existing WebSphere MQ environments.
- Potential for highly available configurations.

11.4.2 Runtime components

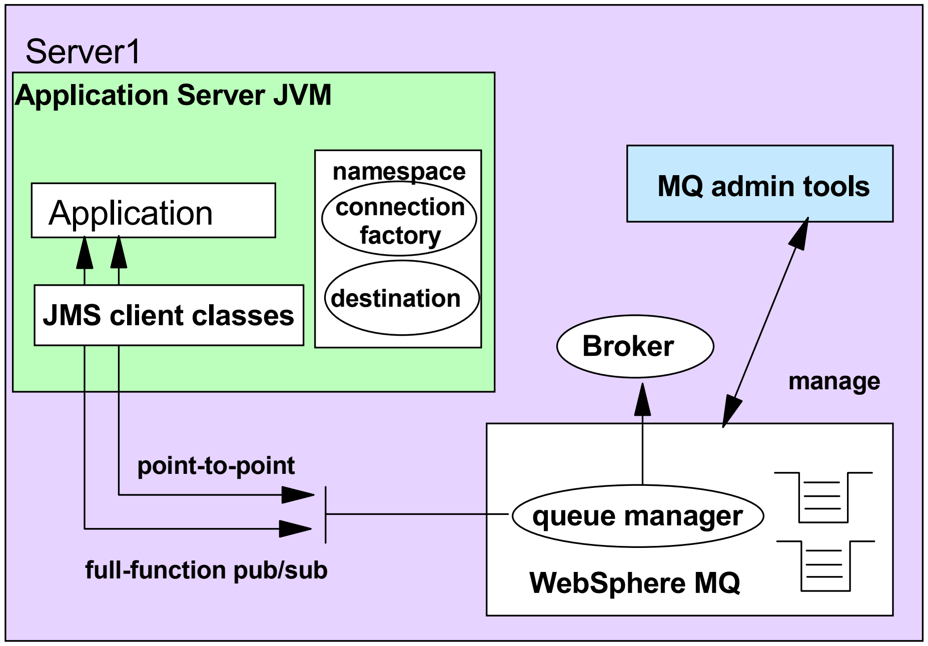


Figure 11-7 Full WebSphere MQ JMS provider components

The key points of the WebSphere MQ JMS provider configuration are:

- The broker is not hosted by the application server or the separate JMS server process. Instead it is a separate process configured into the WebSphere MQ queue manager.
- The JMS server is not used or required. The management of the queue manager, channels, etc., is performed using the WebSphere MQ native tools. The WebSphere administration tools do not support configuration or management of these resources.
- The WebSphere administration tools are used to configure and manage the JMS administered objects (JMS connection factory and JMS destination) in the local name space of the application servers that access WebSphere MQ.
- Either client (local or remote) or bind (local only) WebSphere MQ access mode can be used for JMS client access.
- The full set of WebSphere MQ functionality is available for use, including:
  - Store and forward
  - Queue clustering
  - Support for different brokers

11.4.3 Administration

The WebSphere MQ JMS provider is partially integrated into WebSphere system management. WebSphere administration tools can be used to both configure and manage WebSphere MQ JMS administered objects.

Creation and management of queue managers, channels, and queues must be performed using WebSphere MQ native tools.

11.4.4 Security considerations

The WebSphere MQ JMS provider does not provide functionality to link to the WebSphere security infrastructure.

11.4.5 Interoperability

Interoperates with other non-WebSphere related MQ installations, allowing WebSphere solutions to be integrated into heterogeneous WebSphere MQ environments.

11.5 Generic JMS provider

The generic JMS provider is recommended in the following cases:

- A non-WebSphere MQ messaging system already exists in the environment, and into which the WebSphere installation is required to integrate directly.
- Where a non-WebSphere MQ JMS provider supports functionality that is not available using WebSphere MQ, and which would be useful for the user's messaging environment.
- Connecting clients in one WebSphere Application Server V5 to JMS servers in another WebSphere Application Server V5 cell, in order to enable inter-cell messaging.

11.5.1 Key features

Generic JMS providers have the following key features:

- Not limited to WebSphere MQ messaging systems. This may allow integration into non-WebSphere MQ based environments.
- Provides local JNDI aliases for the real JMS connection factory and JMS destination objects configured into the external name space, hiding the use of an external provider and its resources from the application.

11.5.2 Limitations

Generic JMS providers suffer from the following limitations when used with WebSphere:

- Resources cannot be managed using WebSphere administration tools.
- Not tightly integrated into the WebSphere administration interface or service APIs.
- Connection classes may not support two-phase commit (XA) transactions linked to the WebSphere transaction service.
- May require applications to use non-standard protocols and classes to connect to a remote JNDI name space when accessing the JMS administered objects. The provider may not provide JMS tools that can register its objects in the WebSphere name space.

11.5.3 Runtime components

The key points of the generic JMS provider configuration are:

- Applications reference local JNDI entries set up as aliases to the remote JMS destination and JMS connection factory. It is the responsibility of the WebSphere runtime to resolve the accesses to the actual remote JNDI entries.

  The advantage of this approach is that it provides another level of indirection that protects the application from knowledge of the messaging topology.
- The WebSphere runtime accesses the remote JNDI name space using the provider-specific initial context factory class and provider URL.
- The JMS administered objects in the external JNDI name space are registered using tool(s) provided by the generic JMS provider.
- The actual messaging objects (queues and topics) are created and managed using the provider’s native tools.

See Figure 11-3 on page 462 for an overview of the components involved in WebSphere support for generic JMS providers.

Note: In order for an application to perform a lookup using the WebSphere local JNDI aliases, the generic provider’s initial context factory class and JMS client classes must be in the application’s classpath or the classpath of the application server in which the application is hosted.

11.5.4 Administration

The WebSphere administration tools only provide support for storing the information necessary to connect to the provider's JNDI context root:

- Initial context factory class name
- Provider URL

All configuration and management of the provider's JMS administered objects (JMS connection factory and JMS destination) and messaging resources (queues and topics) must be performed using the provider's own tools.

11.5.5 Security considerations

Generic JMS providers will not link to the WebSphere security infrastructure. This means that any authentication and authorization of JMS clients will have to be provided by the JMS provider itself, separate from WebSphere.

11.5.6 Interoperability

The interoperability of third-party (generic) JMS providers with other providers (WebSphere MQ and others) is dependent upon the underlying message system, protocol, and architecture used by the provider.

If the WebSphere MQ messaging protocol and message format are not used, then the provider will not interoperate directly with WebSphere MQ.

11.6 WebSphere clusters and MQ clusters

This section covers the use of WebSphere horizontal server clusters with WebSphere MQ server clustering. It describes a scenario that shows how the message listener service can be configured to take advantage of WebSphere MQ server clustering and provides some information about how to resolve potential runtime failures in the clustering scenario.

Note: WebSphere MQ server clustering is only available when the full WebSphere MQ product is used as the JMS provider.

For a WebSphere application server configured to use the message listener service, each JMS listener is used to retrieve messages from destinations defined to the server.

In Figure 11-8, the listener configurations are the same for each WebSphere application server. Each application server host contains a WebSphere application server and a WebSphere MQ server. If a host is only used to distribute messages, it only contains a WebSphere MQ server. There can be many servers defined in the configuration, although for simplicity the information in this topic is based on a scenario containing only three servers.

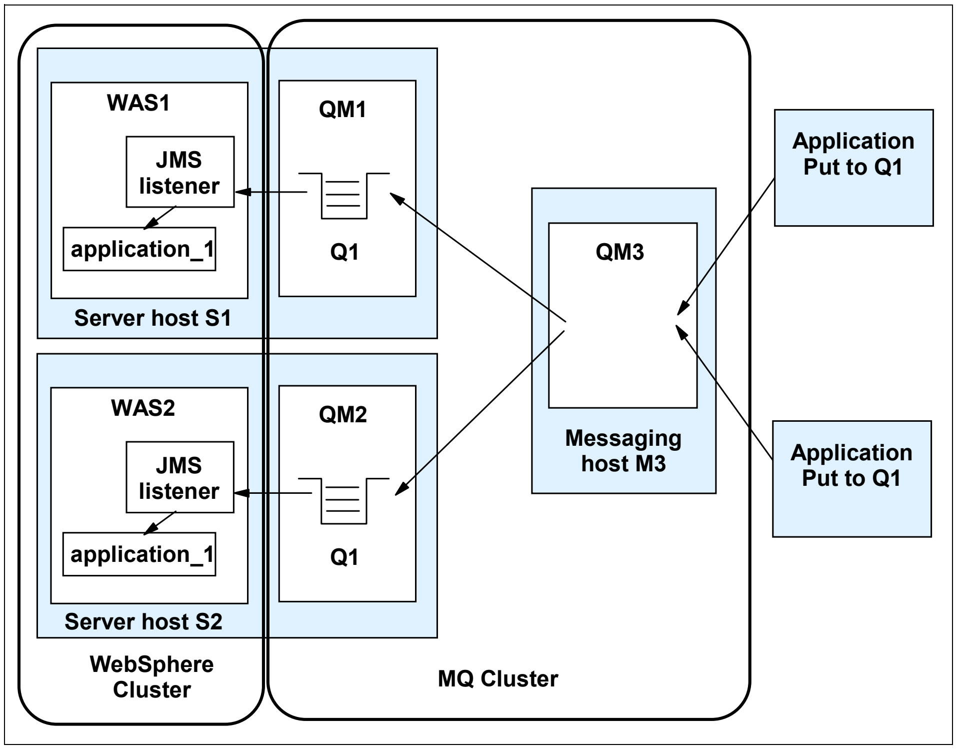


Figure 11-8 WebSphere MQ clustering and WebSphere horizontal cluster

Components

Figure 11-8 shows two WebSphere hosts, with a horizontal cluster, and a messaging host used to distribute messages for WebSphere MQ server clustering.

The scenario comprises the following three hosts:

1. Server host S1 contains the following servers:
   a. WebSphere MQ server

      The server is defined to have a queue manager, QM1, and a local queue, Q1. The queue manager belongs to a cluster. The queue is populated by the WebSphere MQ server located on host M3. Applications can add messages directly to the queue, Q1, but would not be subjected to the control of the WebSphere MQ cluster.

   b. WebSphere

      This contains a member of the horizontal server cluster, WAS1, which is configured with a JMS listener. The listener is configured to retrieve messages from JMS destination Q1.

2. Server host S2 contains the following servers:

   a. WebSphere MQ server

      The server is defined to have a queue manager, QM2, and a local queue, Q1. The queue manager belongs to the same cluster as QM1 on host S1. As with QM1, the queue is populated by the WebSphere MQ server located on host M3. Applications can add messages directly to the queue, Q1, but would not be subjected to the control of the WebSphere MQ cluster.

   b. WebSphere

      This contains a member of the horizontal server cluster, WAS2, which is configured with a JMS listener. The listener is configured to retrieve messages from JMS destination Q1.

3. Messaging host M3 contains the following servers:

   a. WebSphere MQ server

      The server is defined to have a queue manager, QM3, which also belongs to the same cluster as QM1 and QM2. Applications add messages to the queue manager and queue Q1. The cluster to which this queue manager belongs causes messages to be distributed to all other queue managers in the cluster that have queue Q1 defined.

Note:

1. Queue Q1 is not defined as a local queue on host M3. If the queue was defined locally, then messages would remain on the server for local processing; messages would not be distributed by the queue manager cluster control to the other queue managers in the cluster that do have the queue defined.
2. Host M3 does not have an application server defined. All message retrieval processing is performed by the other two application servers on hosts S1 and S2.

Recovery scenarios

There are several failure scenarios that could occur with this clustering configuration. For example:

- Application server failures

  In this scenario the failure of any single WebSphere application server results in the messages for the specified destination remaining on the queue, until the server is restarted.

- WebSphere MQ queue manager failures

  There are two different failures to consider:

  a. Failure of a queue manager on the same host as a WebSphere application server (for example, failure of QM2 on host S2). In this case messages are delivered to the other available application servers, until the WebSphere MQ server is back online, when messages are processed as expected.

  b. Failure of the messaging host M3 and its queue manager, QM3. In this case, the result of an outage is more significant because no messages are delivered to the other queue managers in the cluster. In a fully deployed and scaled production system, host M3 would not be designed to be a single point of failure, and additional messaging servers would be added to the MQ cluster configuration.

11.7 JMS scenarios

WebSphere V5 support for different JMS providers includes a number of choices with which to implement an asynchronous messaging system. Table 11-5 on page 487 provides a summary of the JMS functions supported by each of the scenarios. The decision on which scenario is best suited to a particular JMS messaging solution should be made by matching the scenario that best provides the required functions.

Table 11-5 JMS scenarios versus messaging functions

Function	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Point-to-point	Y	Y	Y	Y	Y	Y	Y	Y
Persistent publish/subscribe	Y[2]	Y[2]	Y[2]	Y[2]	Y[2]	Y	Y	Y
Durable publish/subscribe	Y[2]	Y[2]	Y[2]	Y[2]	Y[2]	Y	Y	Y
Queue clustering	N	N	N	N	N	N	Y	Y
Store and forward	N	N	N	N	N	Y[1]	Y[1]	Y[1]
Shared topic space	N	N	N	N	N	N	Y	Y
JMS server failover	N	N	N	N	N	N	Y	Y
High availability (HA) JMS server	N	N	N	Y	N	N	Y	Y

Scenarios:
1. WebSphere JMS provider - single base server
2. WebSphere JMS provider- multiple base servers
3. WebSphere JMS provider - Network Deployment within a cell
4. WebSphere JMS provider - Network Deployment with high availability
5. WebSphere JMS provider - Network Deployment between cells
6. WebSphere MQ JMS provider - no clustered queues
7. WebSphere MQ JMS provider - clustered queues
8. WebSphere MQ JMS provider - Network Deployment between cells

Notes:

[1]Only if a local queue manager is present.
[2]Only if full-function (queued) publish/subscribe is used.

- Persistent publish/subscribe

  All messages are delivered to a subscriber in order. No messages are lost, even in high-volume situations.

- Durable publish/subscribe

  A subscriber can specify a client ID in their JMS connection and then create a durable subscriber. This causes the JMS server to store persistently all messages matching the subscription on the server. If the client disconnects and then reconnects with the same client ID, then it will start receiving messages from where it left off.

- Shared topic space

  A publisher can publish a message on a topic to a JMS server. Subscribers can connect to another JMS server, subscribe to that topic and receive the message.
- Queue clustering

  Allows the sending of messages to a single logical queue to be load balanced across a number of "clustered" queues.
- Store and forward

  Logging of messages to a database so that they can be forwarded to a remote JMS server at a later time.

11.7.1 WebSphere JMS provider - single base server

The key features of scenario 1 are shown in Figure 11-9.

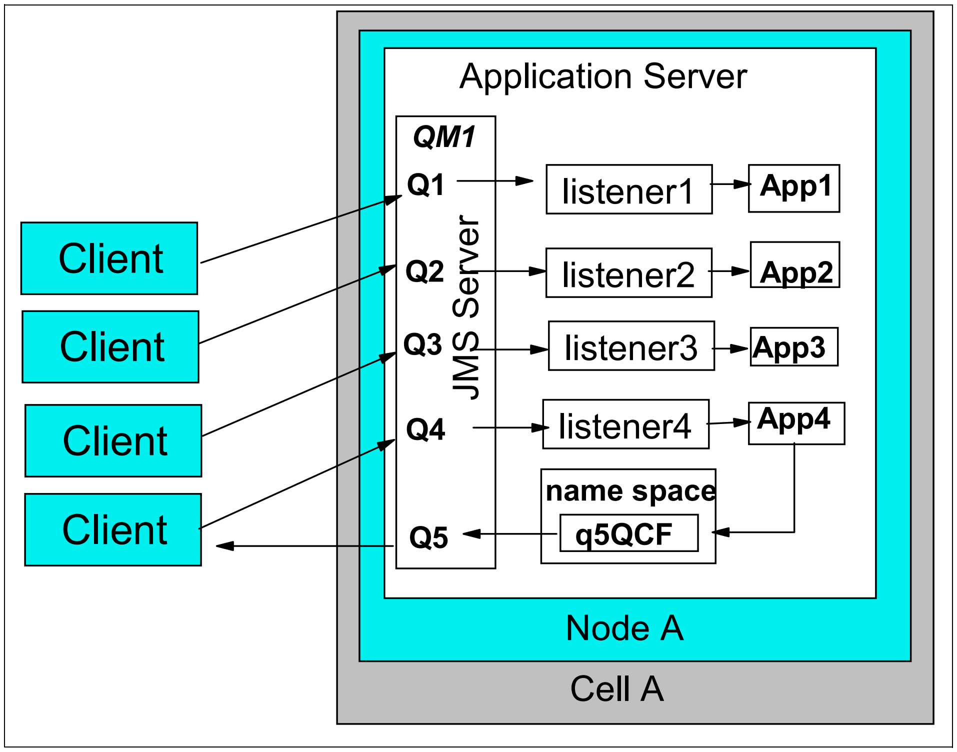


Figure 11-9 WebSphere JMS provider - single base server

Advantages

This scenario has the following advantages:

- No need for a separate installation of WebSphere MQ.
- Administration of messaging resources can be performed using WebSphere administration tools. As a result there is no need to learn to configure WebSphere MQ queue managers, channels, queues, topics and other resources using native WebSphere MQ tools.
- WebSphere security can be enabled to provide authentication and authorization of JMS resources.
- All applications access JMS resources from the application server's local name space.

Limitations

This scenario has the following limitations:

- No JMS server failover or high availability.
- No shared topic space or store and forward.
- No queue clustering.
- Many WebSphere MQ configuration settings are only accessible through manual editing of the WebSphere MQ configuration files.
- Applications can only exchange messages with other applications on the same application server.

When to use this scenario

This scenario is appropriate when the following is true:

1. Full-scale WebSphere MQ features are not required.
2. Only WebSphere Application Server base software is available.
3. Application high availability and fault tolerance is not required, so all applications can be hosted on the one application server.
4. Messaging high availability and failover is not required.

11.7.2 WebSphere JMS provider - multiple base servers

The key features of scenario 2 are shown in Figure 11-10 on page 490.

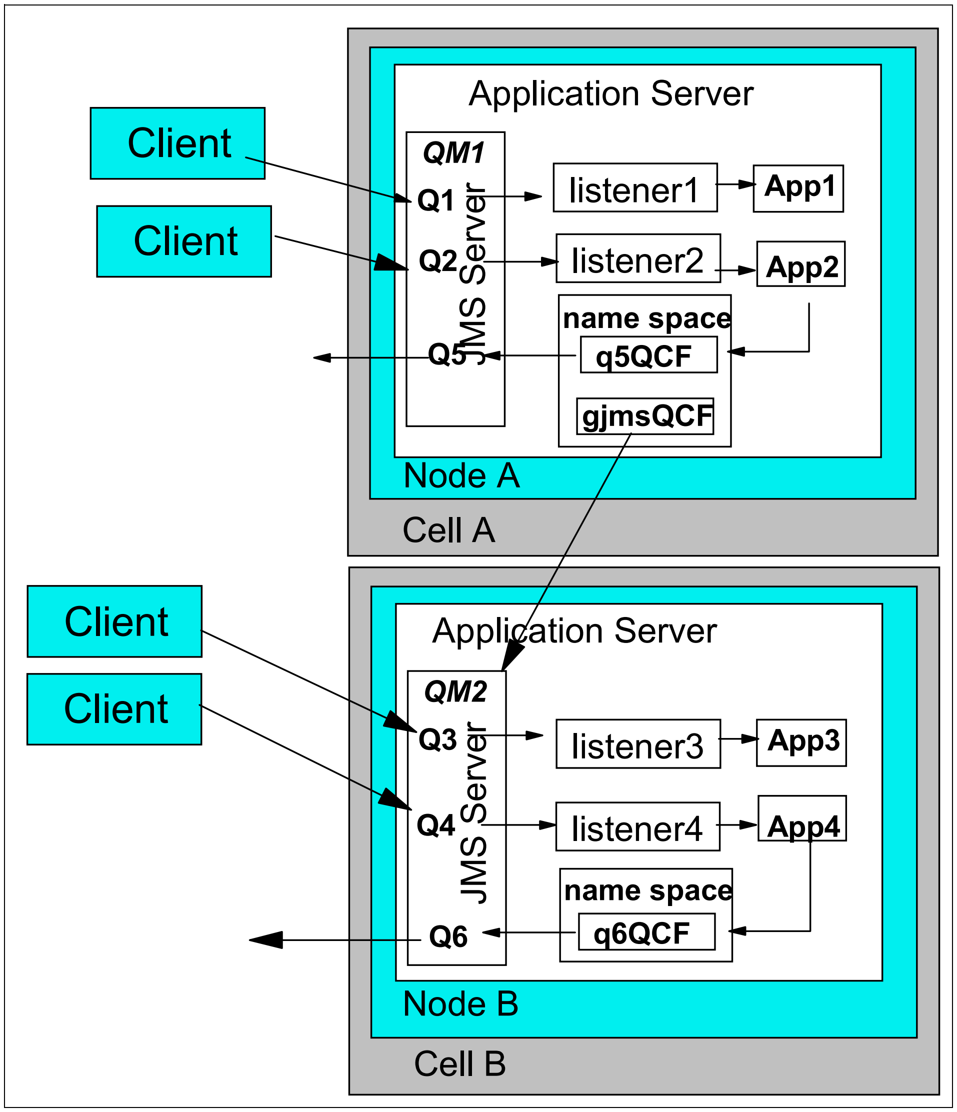


Figure 11-10 WebSphere JMS provider - multiple base servers

Advantages

This scenario has the following advantages:

- No need for a separate installation of WebSphere MQ.
- Administration of messaging resources can be performed using WebSphere administration tools. As a result there is no need to learn to configure

WebSphere MQ queue managers, channels, queues, topics, and other resources using native WebSphere MQ tools.

- WebSphere security can be enabled to provide authentication and authorization of JMS resources.
- All applications do not have to be hosted on the same application server JVM, making the topology less susceptible to the failure of any one application server JVM.

Limitations

This scenario has the following limitations:

- No JMS server failover or high availability.
- No shared topic space or store and forward.
- No queue clustering.
- Many WebSphere MQ configuration settings are only accessible through manual editing of the WebSphere MQ configuration files.
- In order for an application in one application server to interact with the JMS server on another application server, a generic JMS connection factory must be configured to point at the remote JMS server.
- Each application server has to be administered independently, since each has its own configuration repository and WebSphere administrative console.

When to use this scenario

This scenario is appropriate for those situations where scenario 1 cannot be used because applications cannot all be hosted on the same application server, for example in cases where different applications require different application server configurations.

11.7.3 WebSphere JMS provider - Network Deployment

The key features of scenario 3 are shown in Figure 11-11 on page 492.

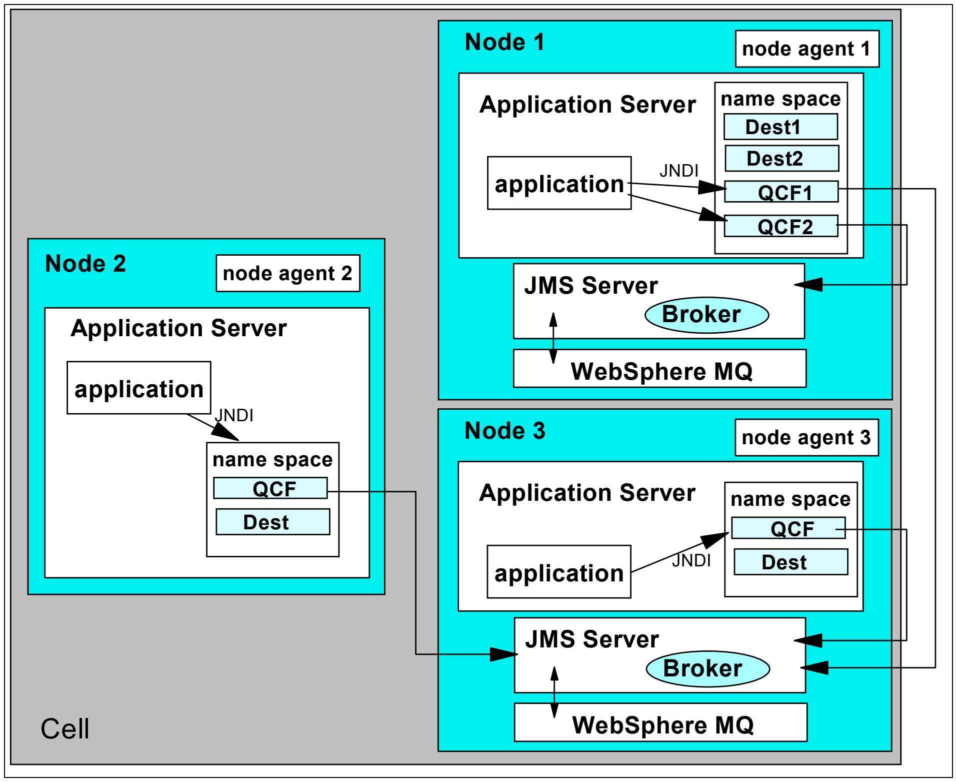


Figure 11-11 WebSphere JMS provider - Network Deployment within a cell

Advantages

This scenario has the following advantages:

- No need for a separate installation of WebSphere MQ.
- Administration of messaging resources can be performed using WebSphere administration tools. As a result there is no need to learn to configure WebSphere MQ queue managers, channels, queues, topics and other resources using native WebSphere MQ tools.
- WebSphere security can be enabled to provide authentication and authorization of JMS resources.
- Applications on any application server in the cell can connect to any JMS server in the cell, using the JMS administered objects registered in the cell's federated name space.
- Each JMS server runs as a separate JVM, enabling it to be started or stopped independent of the application server(s). A node could be configured to not run a JMS server.

- Network Deployment provides high availability and fault tolerance by supporting the configuration of nodes on different machines, as well as the application servers horizontally and/or vertically clustered.

Limitations

This scenario has the following limitations:

- No JMS server failover or high availability.
- No shared topic space or store and forward.
- No queue clustering.
- Many WebSphere MQ configuration settings are only accessible through manual editing of the WebSphere MQ configuration files.
- Requires WebSphere Application Server Network Deployment rather than just the simpler base software.

When to use this scenario

This scenario is appropriate when the following is true:

1. Full-scale WebSphere MQ features are not required.
2. WebSphere Application Server Network Deployment is used to support nodes on separate machines as well as application server clustering. This provides support for application high availability, load balancing, and failover.
3. High availability and failover of the messaging system is not required.

11.7.4 WebSphere JMS provider - single JMS server for a cell

This is a variation of scenario 3. Its key features are shown in Figure 11-12 on page 494.

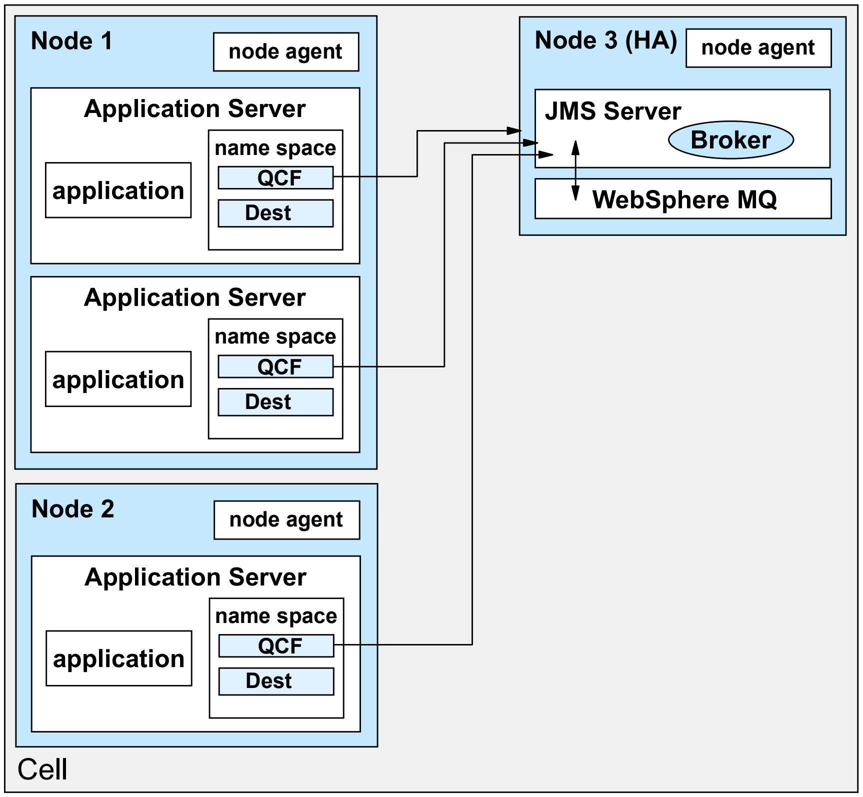


Figure 11-12 WebSphere JMS provider- Network Deployment with high availability

- One of the nodes is dedicated to running the JMS server. That is, it is not used to run application server processes.
- All other nodes do not run a local JMS server.
- The applications hosted by the application servers on all other nodes of the cell use the single dedicated JMS server.
- The machine used for the dedicated JMS is a high-availability configuration.

Advantages

This scenario has the following advantages:

- No need for a separate installation of WebSphere MQ.
- Administration of messaging resources can be performed using WebSphere administration tools. As a result, there is no need to learn to configure WebSphere MQ queue managers, channels, queues, topics and other resources using native WebSphere MQ tools.

- WebSphere security can be enabled to provide authentication and authorization of JMS resources.
- Applications on any application server in the cell can connect to the JMS server.
- The JMS server runs as a separate JVM, enabling it to be started or stopped independent of the application server(s).
- The JMS server’s node is located on a high availability machine so that the JMS server is highly available. Application server’s that do not have to be highly available are located on other machines.
- WebSphere Application Server Network Deployment provides application server high availability and fault tolerance using cluster support. Application servers do not need to be hosted on high availability machines in order to achieve high availability for application servers and their applications.

Limitations

This scenario has the following limitations:

- No shared topic space or store and forward.
- No queue clustering.
- Many WebSphere MQ configuration settings are only accessible through manual editing of the WebSphere MQ configuration files.
- Requires Network Deployment rather than just the simpler base software.
- Requires a high-availability (hardware and software) machine for hosting the JMS server. Such high-availability machines can be expensive.

When to use this scenario

This scenario is appropriate for those situations where scenario 3 cannot be used because high availability of the messaging system is required.

11.7.5 WebSphere JMS provider - Network Deployment between cells

The key features of scenario 5 are shown in Figure 11-13 on page 496.

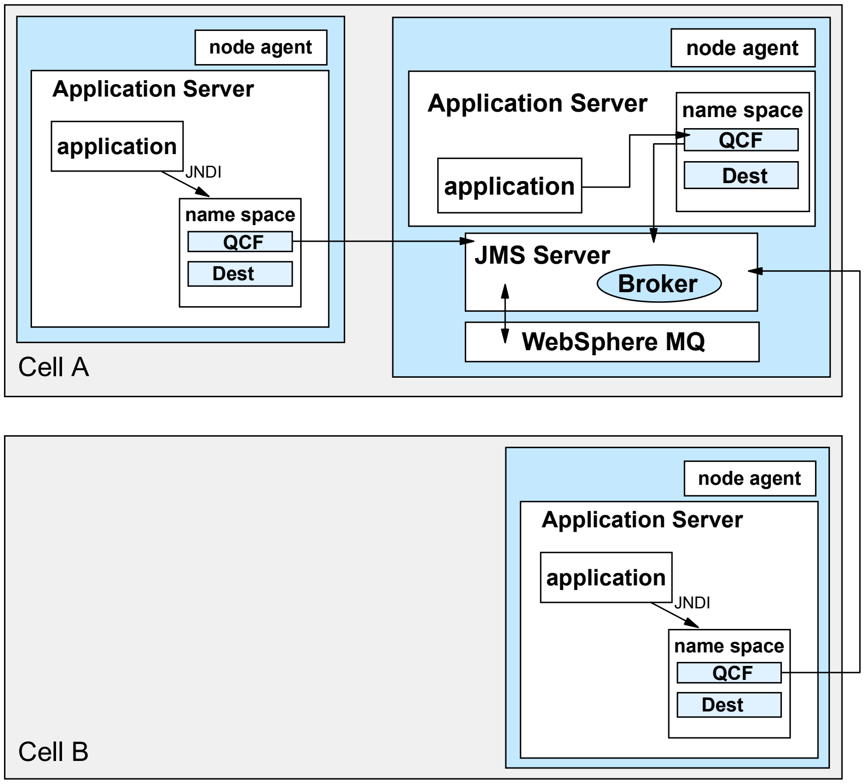


Figure 11-13 WebSphere JMS provider- Network Deployment between cells

- Applications in each cell can communicate with JMS servers in the same cell by using the WebSphere JMS provider resources in the cell name space (via JNDI).
- In order for an application in one cell to interact with a JMS server in another cell, a *generic JMS connection factory* must be configured in its cell to point at the remote JMS server.

Advantages

This scenario has the following advantages:

- No need for a separate installation of WebSphere MQ.
- Administration of messaging resources can be performed using WebSphere administration tools. As a result there is no need to learn to configure WebSphere MQ queue managers, channels, queues, topics and other resources using native WebSphere MQ tools.

- WebSphere security can be enabled to provide authentication and authorization of JMS resources.
- Applications on any application server in the cell can connect to the JMS server.
- The JMS server runs as a separate JVM, enabling it to be started or stopped independent of the application server(s).
- Network Deployment provides application server high availability and fault tolerance using cluster support. Application servers do not need to be hosted on high-availability machines in order to achieve high availability for application servers and their applications.

Limitations

This scenario has the following limitations:

- No JMS server failover or high availability.
- No shared topic space or store and forward.
- No queue clustering.
- Many WebSphere MQ configuration settings are only accessible through manual editing of the WebSphere MQ configuration files.
- Requires Network Deployment rather than just the simpler base software.

When to use this scenario

This scenario is appropriate for those situations where scenario 3 cannot be used because applications in multiple WebSphere V5 cells must be able to exchange messages, but where the features of full-scale WebSphere MQ are not required.

11.7.6 WebSphere MQ JMS - no clustered queues

The key features of scenario 6 are shown in Figure 11-14 on page 498.

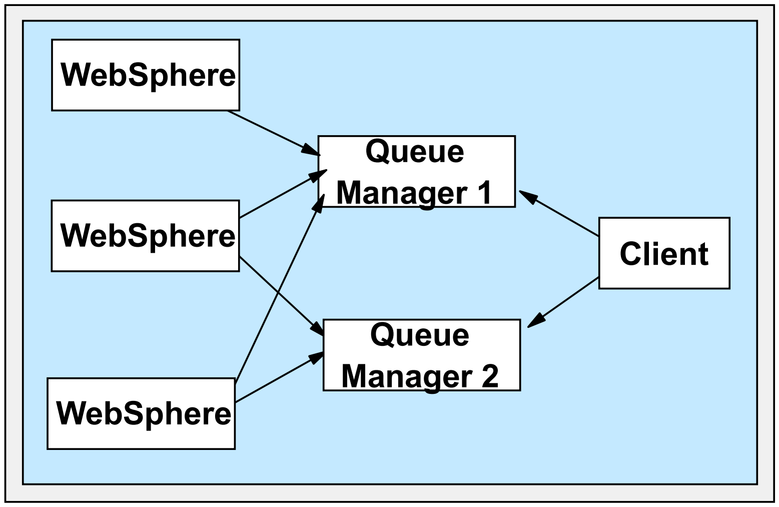


Figure 11-14 Full WebSphere MQ - no clustered queues

- If an application has a local queue manager, then store and forward is possible. WebSphere MQ communications using client mode (non-bind) cannot perform store and forward.
- Shared topic space.
- Each queue is independent and distinct.

Advantages

This scenario has the following advantages:

- The features of full-scale WebSphere MQ can be used, for example queue clustering, shared topic spaces, message store and forward.

Limitations

This scenario has the following limitations:

- Administration of WebSphere MQ resources (queues, topics) cannot be performed using the WebSphere administration tools. Such administration must be performed using native WebSphere MQ tools.
- Requires separate purchase and installation of full-scale WebSphere MQ.

When to use this scenario

This scenario is appropriate for those situations where the WebSphere MQ features not provided by the WebSphere JMS provider, except queue clustering, are required.

11.7.7 WebSphere MQ JMS - clustered queues

The key features of scenario 7 are shown in Figure 11-15.

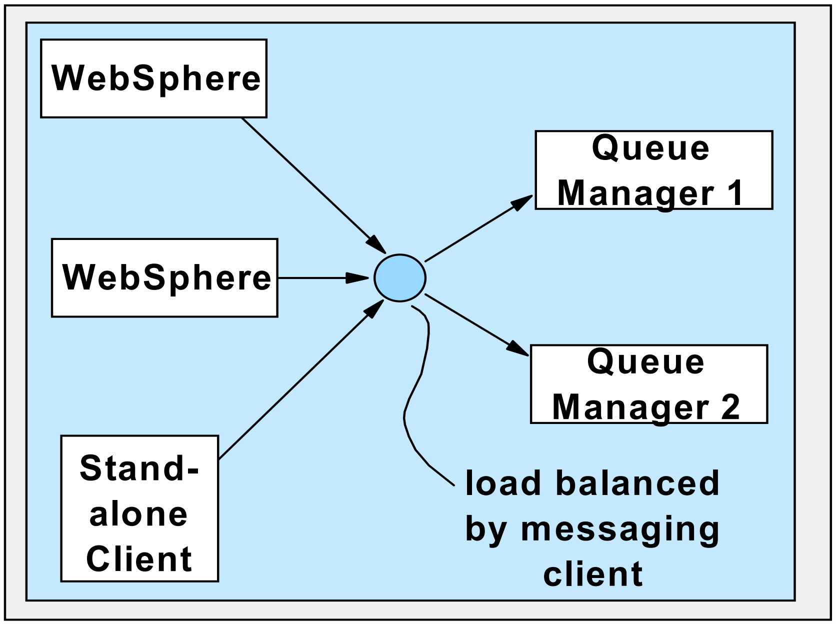


Figure 11-15 Full WebSphere MQ - clustered queues

- Clustering is performed at the client end. One or more queue managers are repositories keeping track of the cluster status.
- A client asks a repository for all queue managers hosting a particular clustered queue, and then the client load balances PUT requests between them.
- GET requests are not clustered. A client attaches to a queue on a queue manager and gets messages from it.

Advantages

This scenario has the following advantages:

- The features of full-scale WebSphere MQ can be used. For example, queue clustering, shared topic spaces, message store and forward.

Limitations

This scenario has the following limitations:

- Administration of WebSphere MQ resources (queues, topics) cannot be performed using the WebSphere administration tools. Such administration must be performed using native WebSphere MQ tools.
- Requires separate purchase and installation of full-scale WebSphere MQ.

When to use this scenario

This scenario is appropriate for those situations where scenario 6 cannot be used because queue clustering is required.

11.7.8 WebSphere MQ JMS - Network Deployment between cells

The key features of scenario 8 are shown in Figure 11-16 on page 501.

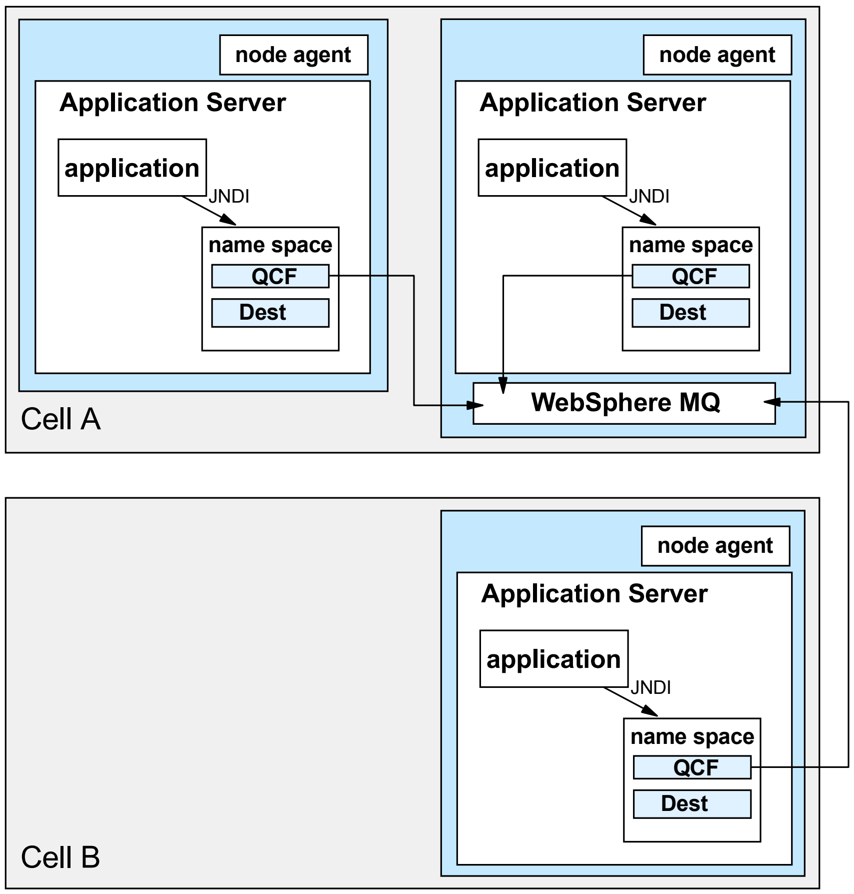


Figure 11-16 Full WebSphere MQ - Network Deployment between cells

Advantages

This scenario has the following advantages:

- The features of full-scale WebSphere MQ can be used, for example queue clustering, shared topic spaces, message store and forward.
- Shared topic space is possible between cells.
- Store and forward is possible between cells if the node sending the message has a local WebSphere MQ queue manager.
- A client can use queue clustering to spread messages over two cells. This enables one cell to back up the operation of the other.

Limitations

This scenario has the following limitations:

- Administration of WebSphere MQ resources (queues, topics) cannot be performed using the WebSphere administration tools. Such administration must be performed using native WebSphere MQ tools.
- Requires separate purchase and installation of full-scale WebSphere MQ.

When to use this scenario

This scenario is appropriate for those situations where scenario 5 cannot be used because the features of full-scale WebSphere MQ are required.

11.8 Configuration and management

The WebSphere administrative console can be used to configure and administer the following for JMS:

- JMS providers

  WebSphere Application Server V5 offers three choices when selecting a JMS provider. During installation, you can choose to install the embedded WebSphere JMS provider. If you elect to use an external provider instead, you can use WebSphere MQ, or you can install a generic JMS provider. The WebSphere JMS provider and WebSphere MQ provider are pre-configured. If you use a generic provider, you need to define it to WebSphere.

- JMS resources

  JMS resources are the key to JMS portability. JMS resources are created by a JMS administrator and are specific to the underlying JMS provider implementation. These JMS resources implement standard JMS interfaces and are retrieved by JMS applications through JNDI. JMS developers must know only the JNDI name of the JMS resources, and do not have to write any vendor-specific code to use the administered objects.

  If you use the WebSphere JMS provider or the WebSphere MQ provider, these resources can be defined to the WebSphere name space using the WebSphere administrative console. Generic JMS providers must provide a tool to access the WebSphere name space.

11.8.1 Managing the WebSphere JMS provider

The WebSphere JMS provider is pre-configured for you. To view the configuration properties of the embedded WebSphere JMS provider, complete the following steps:

1. In the navigation tree, expand **Resources**.
2. Select **WebSphere JMS Provider**.
3. Set the scope to the node where the WebSphere JMS provider is installed.
4. Click **Apply**.
5. In the General Properties table, you can view the name and description properties for the JMS provider. There is one per node, defined at installation time. These properties cannot be changed.
6. In the Additional Properties table, you can view and change the resources for this JMS provider.

Note: When the embedded WebSphere JMS provider is installed, there are a number of WebSphere MQ properties, configured automatically.

In general, the default values of WebSphere MQ properties are adequate. However, if you are running high messaging loads, you may need to change some WebSphere MQ properties, for example properties for log file locations, file pages, and buffer pages. For more information about configuring WebSphere MQ properties, see *MQSeries System Administration*, SC33-1873.

Managing the WebSphere JMS provider process

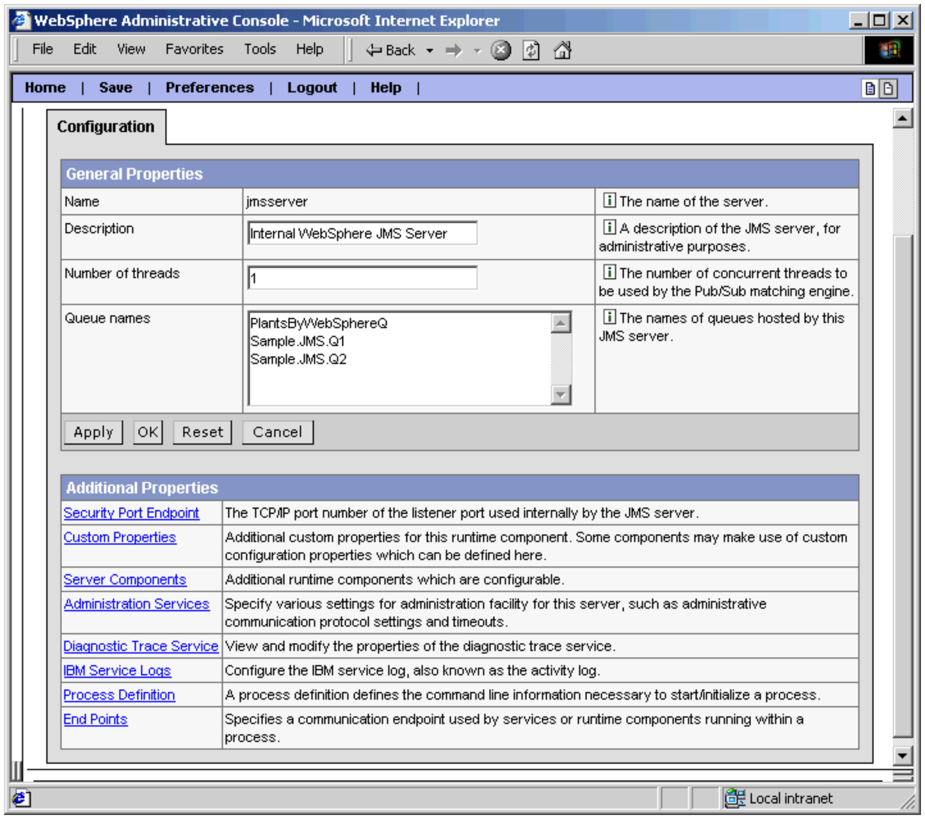
You can use the administrative console to display a list of all the JMS server processes in your cell, and to view and control their runtime status. You can also use the administrative console to configure the general set of JMS server process properties, which add to the property values configured for the WebSphere JMS provider.

To configure the properties of a JMS server process using the administrative console, complete the following steps:

1. In the navigation tree, select **Servers -> JMS Servers**. This displays all the JMS server processes configured in your cell (one per node in which embedded messaging has been installed) and their runtime status.

Base installations only: The JMS server process resides in the application server process. To configure the JMS server on a base installation, select **Servers -> Application Servers**. Click **Server1**. Select **Server Components -> JMS Servers**.

2. Click the jmsserver instance for the node you want to edit to display the properties. Figure 11-17 shows the configuration properties and the information fields that describe each.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Home | Save | Preferences | Logout | Help |
Configuration
Local intranet

General Properties

Name	jmsserver	The name of the server.
Description	Internal WebSphere JMS Server	A description of the JMS server, for administrative purposes.
Number of threads	1	The number of concurrent threads to be used by the Pub/Sub matching engine.
Queue names	PlantsByWebSphereQ Sample.JMS.Q1 Sample.JMS.Q2	The names of queues hosted by this JMS server.

Apply OK Reset Cancel

Additional Properties

Security Port Endpoint	The TCP/IP port number of the listener port used internally by the JMS server.
Custom Properties	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.
Server Components	Additional runtime components which are configurable.
Administration Services	Specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts.
Diagnostic Trace Service	View and modify the properties of the diagnostic trace service.
IBM Service Logs	Configure the IBM service log, also known as the activity log.
Process Definition	A process definition defines the command line information necessary to start/initialize a process.
End Points	Specifies a communication endpoint used by services or runtime components running within a process.

Figure 11-17 WebSphere JMS provider properties

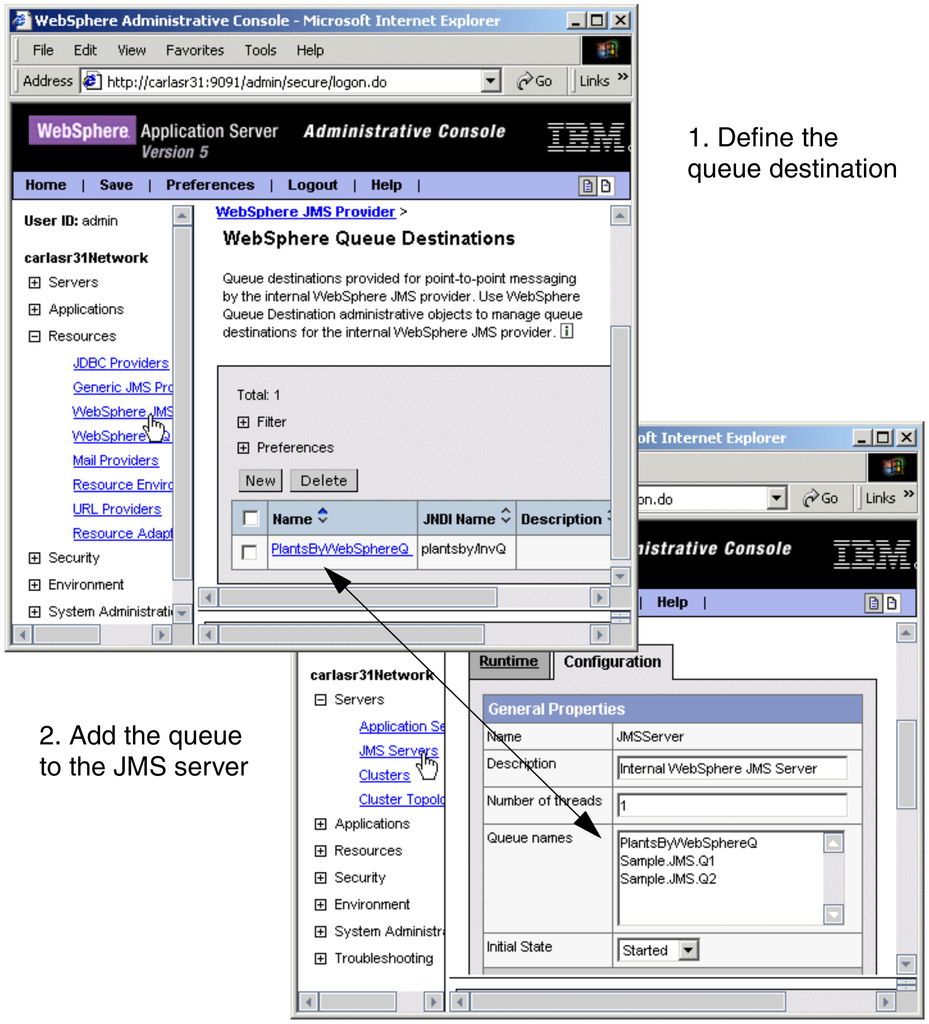
Notes:

– The name of the process is JMSServer and can’t be changed.

– The number of concurrent threads should only be set to a small number. The default value is 1.

– The list of queue names represents the actual messaging queues that get created. Each queue listed in this field must match the name of a WebSphere queue administrative object (including the use of upper and lowercase). By adding the queue name to a specific JMS server process, we are specifying for which WebSphere JMS provider (if there is more than one) the messaging queue gets created.

Figure 11-18 shows this relationship. To make a queue available to an application, you first define the queue destination. Then, you add the destination to the Queue Names field of the JMS server.



Queue destinations provided for point-to-point messaging by the internal WebSphere JMS provider. Use WebSphere Queue Destination administrative objects to manage queue destinations for the internal WebSphere JMS provider.

	Name	JNDI Name	Description
☐	PlantsByWebSphereQ	plantsby/InvQ	

Name	JMSServer
Description	Internal WebSphere JMS Server
Number of threads	1
Queue names	PlantsByWebSphereQ Sample.JMS.Q1 Sample.JMS.Q2
Initial State	Started

Figure 11-18 Making a queue accessible to an application

To remove a queue from the JMS server process, remove its name from the Queue names field.

Note: When a queue is added to a JMS server process, the queue name is added to the list of queues hosted by that JMS server process. The list is defined in the jmsserver's server.xml configuration file.

3. Among the items of interest in the Additional Properties table, you can view and change the following:
   - **Security Port Endpoint:** The TCP/IP port used by the JMS server process when security is enabled. The default value is 5557. The security port number is listed in the server.xml configuration file for jmsserver in a Network Deployment environment and for server1 in a base installation.
   - **Endpoints**
     - **JMS server queued address**

       The TCP/IP port used by the JMS server process to start the JMS provider queue manager, for all point-to-point and full-function publish/subscribe support. The default value is 5558.

       The JMS server queued port number (JMSSERVER_QUEUED_ADDRESS) is listed in the serverindex.xml file for the node hosting the server, under the servername=”jmsserver” stanza.
     - **JMS server direct address**

       The TCP/IP port used by the JMS server process for the default (high performance) publish/subscribe support. The default value is 5559. The JMS server direct port number (JMSSERVER_DIRECT_ADDRESS) is listed in the serverindex.xml file for the node hosting the server, under the servername=”jmsserver” stanza.
     - **Soap Connector Address**

       The SOAP JMX management port used by the JMS server. The default value is 8876. The JMS server SOAP connector address (SOAP_CONNECTOR_ADDRESS) is listed in the serverindex.xml file for the node hosting the server, under the servername=”jmsserver” stanza.
     - **Bootstrap Address**

       The Bootstrap port used by the JMS server. The default value is 2810. The JMS server Bootstrap address (SOAP_CONNECTOR_ADDRESS) is listed in the serverindex.xml file for the node hosting the server, under the servername="jmsserver" stanza.

These port definitions are saved in the serverindex.xml file for the node hosting the server.

<WAS_HOME>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml

Example 11-3 jmsserver port entries in serverindex.xml

```
<serverEntries xmi:id="ServerEntry_3" serverDisplayName="JMSServer" serverName="jmsserver"
serverType="MESSAGE_BROKER">
- <specialEndpoints xmi:id="NamedEndPoint_18" endPointName="BOOTSTRAP_ADDRESS">
  <endPoint xmi:id="EndPoint_16" host="carlasr31" port="2810" />
  </specialEndpoints>
- <specialEndpoints xmi:id="NamedEndPoint_13" endPointName="JMSSERVER_DIRECT_ADDRESS">
  <endPoint xmi:id="EndPoint_17" host="carlasr31" port="5559" />
  </specialEndpoints>
- <specialEndpoints xmi:id="NamedEndPoint_14" endPointName="JMSSERVER_QUEUED_ADDRESS">
  <endPoint xmi:id="EndPoint_18" host="carlasr31" port="5558" />
  </specialEndpoints>
- <specialEndpoints xmi:id="NamedEndPoint_19" endPointName="SOAP_CONNECTOR_ADDRESS">
  <endPoint xmi:id="EndPoint_19" host="carlasr31" port="8876" />
  </specialEndpoints>
```

For more information on these ports, see 11.3.4, "Runtime components" on page 469.

4. When you are done, click **OK** and save the configuration. To have the changes take effect, stop then restart the JMS server process.

To change the runtime status of a JMS server process, complete the following steps (Network Deployment only):

1. In the navigation tree, select **Servers -> JMS Servers**.
2. Check the box to the left of the JMS server process that you want to act on.
3. Click **Start** to start the server, or **Stop** to stop it.

Note: The queue manager cannot be administered. It is started and stopped when you start and stop the JMS server process for the node.

11.8.2 Managing the WebSphere MQ JMS provider

The WebSphere MQ JMS provider is pre-configured. Be sure to check the ${MQJMS_LIB_ROOT} and ${MQ_INSTALL_ROOT} variables to make sure they point to the appropriate location for WebSphere MQ libraries.

If you have installed WebSphere MQ as the JMS provider, over the embedded WebSphere JMS provider, complete the following steps to view the configuration properties of the WebSphere MQ JMS provider:

1. In the navigation tree, expand **Resources -> WebSphere MQ JMS Providers**.
2. Set the scope to the node where the WebSphere MQ JMS provider has been installed.
3. Click **Apply**.
4. In the General Properties table, you will see the following properties:
   - **Name:** The name by which the WebSphere MQ JMS provider is known, for administrative purposes. The default is *WebSphere MQ JMSProvider*.
   - **Description:** A description of the JMS provider, for administrative purposes. The default is *WebSphere MQ JMS Provider*.
   - **Classpath:** A list of paths or JAR file names which together form the location for the resource provider classes. The default is ${MQJMS_LIB_ROOT}, which is set to `${MQ_INSTALL_ROOT}/Java/lib`.

     Select **Environment -> Manage WebSphere Variables** to view the values of these WebSphere variables.
   - **Native Library Path:** An optional path to any native libraries (.dll's, .so's). The default is also $(MQJMS_LIB_ROOT).
5. The Additional Properties table provides links to configuration pages where can view and change the resources for this JMS provider.

11.8.3 Managing a generic JMS provider

If you are using a JMS provider other than the embedded WebSphere JMS provider or the WebSphere MQ JMS provider, you first need to define it to WebSphere Application Server. Using the administrative console, complete the following steps:

1. In the navigation tree, expand **Resources -> Generic JMS Providers**.
2. Select **Scope** and click **Apply**.
3. Click **New** in the content page.
4. Define the JMS provider by specifying the appropriate values in the General Properties table.
   - **Name:** The name by which the JMS provider is known, for administrative purposes.

   - **Description:** A description of the JMS provider, for administrative purposes.
   - **Classpath:** A list of paths or JAR file names which together form the location for the resource provider classes.
   - **Native Library Path:** An optional path to any native libraries (.dll's, .so's).
   - **External initial context factory:** The Java classname of the JMS providers initial context factory.

     For example this would be com.swiftmq.jndi.InitialContextFactoryImpl for the SwiftMQ JMS provider.
   - **External provider URL:** The JMS provider URL for external JNDI lookups.

     This specifies how JNDI lookups should be performed. Continuing with the example above, smqp://localhost:4001 would indicate that JNDI lookups should be performed by a JMS inbound listener on host localhost, port 4001, and smqp is the SwiftMQ Protocol.
5. Click **OK**.
6. Use the Additional Properties table to define the JMS destinations and JMS connection factories.
7. Save your configuration.

11.8.4 Configuring JMS resources

In order to send a JMS message from an application, you need the following JMS resources:

- A connection to the JMS provider. This is made possible by a JMS connection factory object. The JMS connection factory object is used by the application to connect to the JMS provider or message router. The following connection factories are configurable:
  - Queue connection factories
  - Topic connection factories
- A destination address for the message. The destination address of the message is identified by a destination object. This is used by the application to send and receive messages.

  The destination object represents a network-independent destination to which the message will be addressed. In JMS, messages are sent to destinations, instead of directly to other applications.

There are two types of destination objects:

- **Topic destination**

  A topic is analogous to an e-mail list or newsgroup. Any application with the proper credentials can receive messages from and send messages to a topic. When a JMS client receives messages from a topic, the client is said to subscribe to that topic.

- **Queue destination**

  Queues are intended for point-to-point messaging. A queue may have multiple receivers, but only one receiver may receive each message.

The following topics describe how these resources get configured into the WebSphere name space and how these resources get created for the JMS provider.

- "Configuring resources for the WebSphere JMS provider" on page 510
- "Configuring resources for WebSphere MQ JMS provider" on page 524
- "Configuring resources for generic JMS provider" on page 543

11.8.5 Configuring resources for the WebSphere JMS provider

If you are using the WebSphere JMS provider in your environment, use the information in this section to configure connection factories and destinations into the WebSphere name space.

Configuring a queue connection factory

A queue connection factory object is used to create JMS connections to the JMS provider for point-to-point messaging with the JMS provider queues (WebSphere MQ queues). Connection factory properties control how connections are created to the associated JMS provider and the underlying queue destinations.

To configure properties for a queue connection factory object used to connect to a WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **Resources -> WebSphere JMS Provider**.
2. Select **Scope** and click **Apply**.

   JMS resources are created at a scope level. When a JMS resource is created, the resources.xml file for that level gets updated with the new resource.

   Depending on the scope level selected, the resource will be available to one or more application servers. It is the application server's JNDI name space that gets updated with the new resource.

For example, if we create a JMS resource such as a queue destination at the node level, this is what happens:

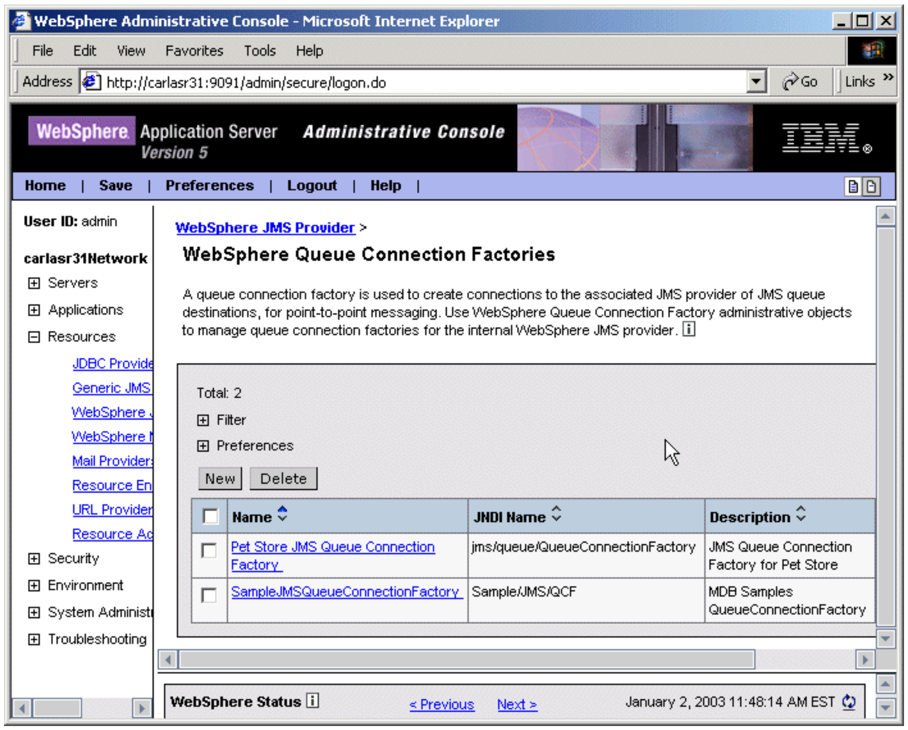
a. The resources.xml file at the node level gets updated with the new queue.

b. The local name space of each application server (but not the node agent) running under that node gets updated with the new resource.

c. Applications running under any of the application servers in that node can now use the resource, either from the local application server name space or from another server's name space using the federated structure of the cell name space.

Note: The resources.xml file is of *merged* type scope. See Chapter 7, "Administration overview" on page 205 for more information on scope types.

3. Click **WebSphere Queue Connection Factories** in the Additional Properties table. This is shown in Figure 11-19.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9091/admin/secure/logon.do
Go Links
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Network
Servers
Applications
Resources
JDBC Provide
Generic JMS
WebSphere J
WebSphere M
Mail Provider
Resource En
URL Provider
Resource Ad
Security
Environment
System Administ
Troubleshooting
WebSphere Status
< Previous
Next >
January 2, 2003 11:48:14 AM EST

WebSphere JMS Provider >

WebSphere Queue Connection Factories

A queue connection factory is used to create connections to the associated JMS provider of JMS queue destinations, for point-to-point messaging. Use WebSphere Queue Connection Factory administrative objects to manage queue connection factories for the internal WebSphere JMS provider.

Total: 2

Filter

Preferences

New Delete

	Name	JNDI Name	Description
☐	Pet Store JMS Queue Connection Factory	jms/queue/QueueConnectionFactory	JMS Queue Connection Factory for Pet Store
☐	SampleJMSQueueConnectionFactory	Sample/JMS/QCF	MDB Samples QueueConnectionFactory

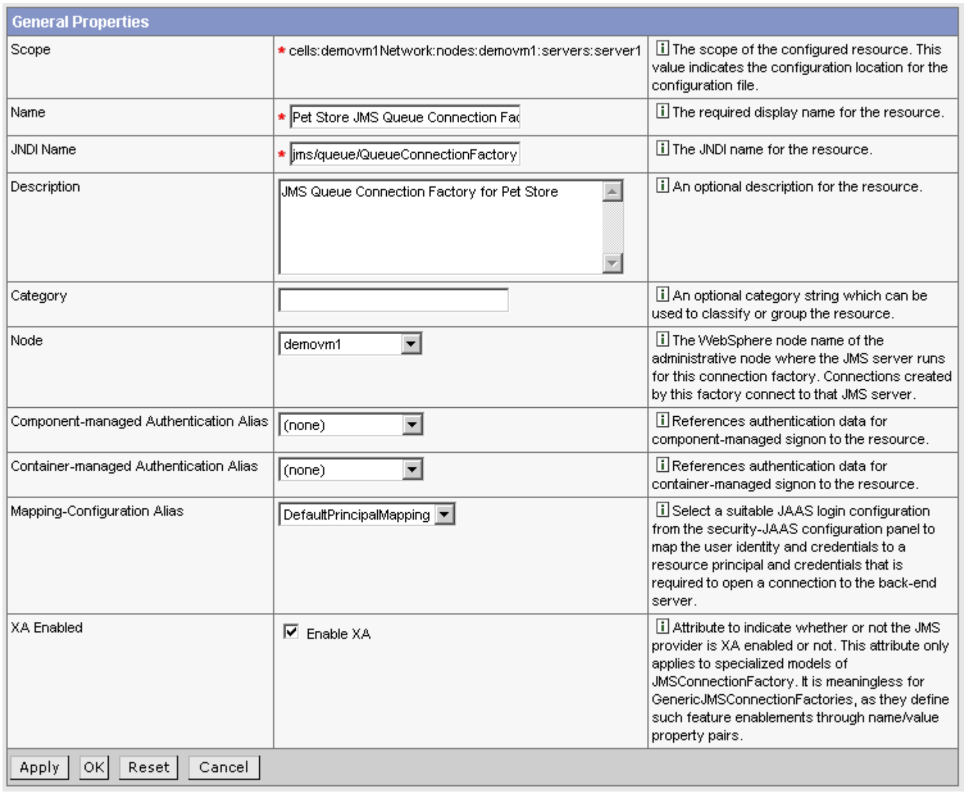
Figure 11-19 Queue connection factory administered objects at the selected scope

In this example we have two WebSphere queue connection factory objects, Pet Store JMS Queue Connection Factory and SampleJMSQueueConnectionFactory. These two queue connection factories have the properties to connect to an embedded WebSphere JMS provider in the cell.

4. To create a new queue connection factory object, click **New** in the content pane. See Figure 11-19 on page 511.

   To change the properties of an existing queue connection factory, click one of the connection factories displayed.

   Figure 11-20 shows the properties for the Pet Store JMS Queue Connection Factory queue connection factory.



General Properties

Scope	* cells:demovm1Network:nodes:demovm1:servers:server1	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* Pet Store JMS Queue Connection Fac	The required display name for the resource.
JNDI Name	* jms/queue/QueueConnectionFactory	The JNDI name for the resource.
Description	JMS Queue Connection Factory for Pet Store	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Node	demovm1	The WebSphere node name of the administrative node where the JMS server runs for this connection factory. Connections created by this factory connect to that JMS server.
Component-managed Authentication Alias	(none)	References authentication data for component-managed signon to the resource.
Container-managed Authentication Alias	(none)	References authentication data for container-managed signon to the resource.
Mapping-Configuration Alias	DefaultPrincipalMapping	Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and credentials to a resource principal and credentials that is required to open a connection to the back-end server.
XA Enabled	✓ Enable XA	Attribute to indicate whether or not the JMS provider is XA enabled or not. This attribute only applies to specialized models of JMSConnectionFactory. It is meaningless for GenericJMSConnectionFactories, as they define such feature enablements through name/value property pairs.

Figure 11-20 Queue connection factory properties

The configuration properties to note are:

– **JNDI name:** Used to bind the connection factory into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

- **Node:** The node name of the administrative node where the embedded WebSphere JMS provider for this connection factory runs. Connections created by this factory connect to that provider.
- **Component-managed/Container-managed Authentication Alias:** Specifies a user ID and password to be used to authenticate connection to a JMS provider. The entry references authentication data defined in the J2C authentication data entries (in the Additional Properties table).

Note: The node name is the WebSphere logical node name, not the host name of the machine.

5. The following properties are also available for configuring:
   - Connection pool settings
   - Session pool settings
   - J2C authentication entries.

   Review the settings and change if appropriate.

   Note: JMS connection and session resources are managed by the J2C connection manager.

6. Click **OK** when finished and save your configuration.
7. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configuring topic connection factories

A topic connection factory object is used to create JMS connections to a JMS publish/subscribe provider. Topic connection factory objects are configured into the application server's name space using the administrative console. The topic connection factory properties control how connections are created to the associated provider and the underlying topic destinations.

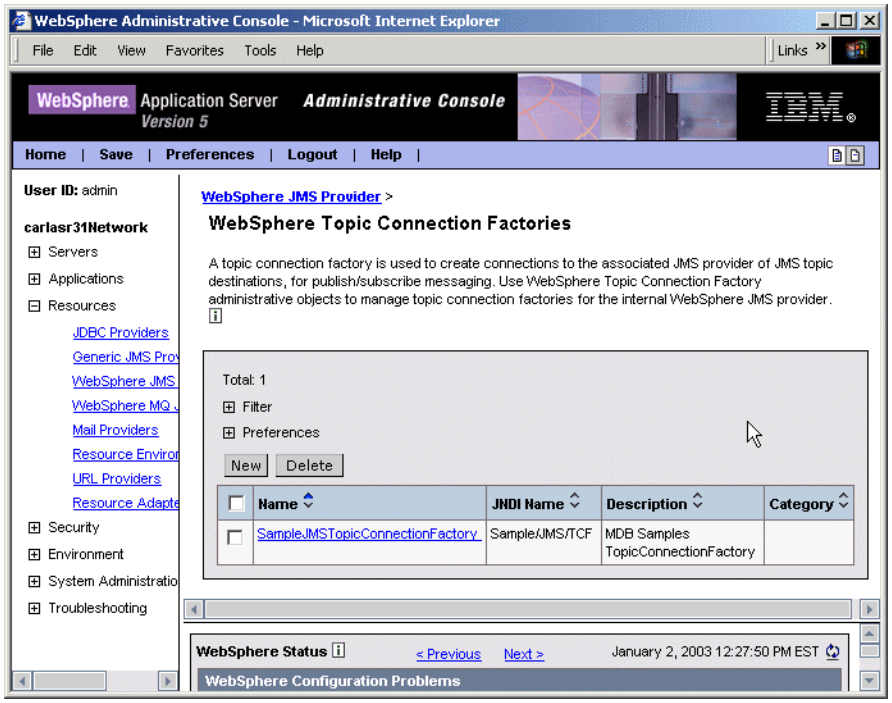
To configure the properties of a topic connection factory object used to connect to the publish/subscribe provider installed with the embedded WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **Resources -> WebSphere JMS Providers**.
2. Select **Scope** and click **Apply**.

   For a discussion on the scope setting, see "Configuring a queue connection factory" on page 510.

3. Click **WebSphere Topic Connection Factories** in the Additional Properties table.

   This displays WebSphere topic connection factory administered objects available to application servers under the selected scope, which are used to connect to a publish/subscribe JMS server installed with an embedded WebSphere JMS provider. This is shown in Figure 11-21.



WebSphere Administrative Console - Microsoft Internet Explorer

WebSphere JMS Provider >

WebSphere Topic Connection Factories

A topic connection factory is used to create connections to the associated JMS provider of JMS topic destinations, for publish/subscribe messaging. Use WebSphere Topic Connection Factory administrative objects to manage topic connection factories for the internal WebSphere JMS provider.

Total: 1

	Name	JNDI Name	Description	Category
☐	SampleJMSTopicConnectionFactory	Sample/JMS/TCF	MDB Samples TopicConnectionFactory	

WebSphere Status | < Previous | Next > | January 2, 2003 12:27:50 PM EST

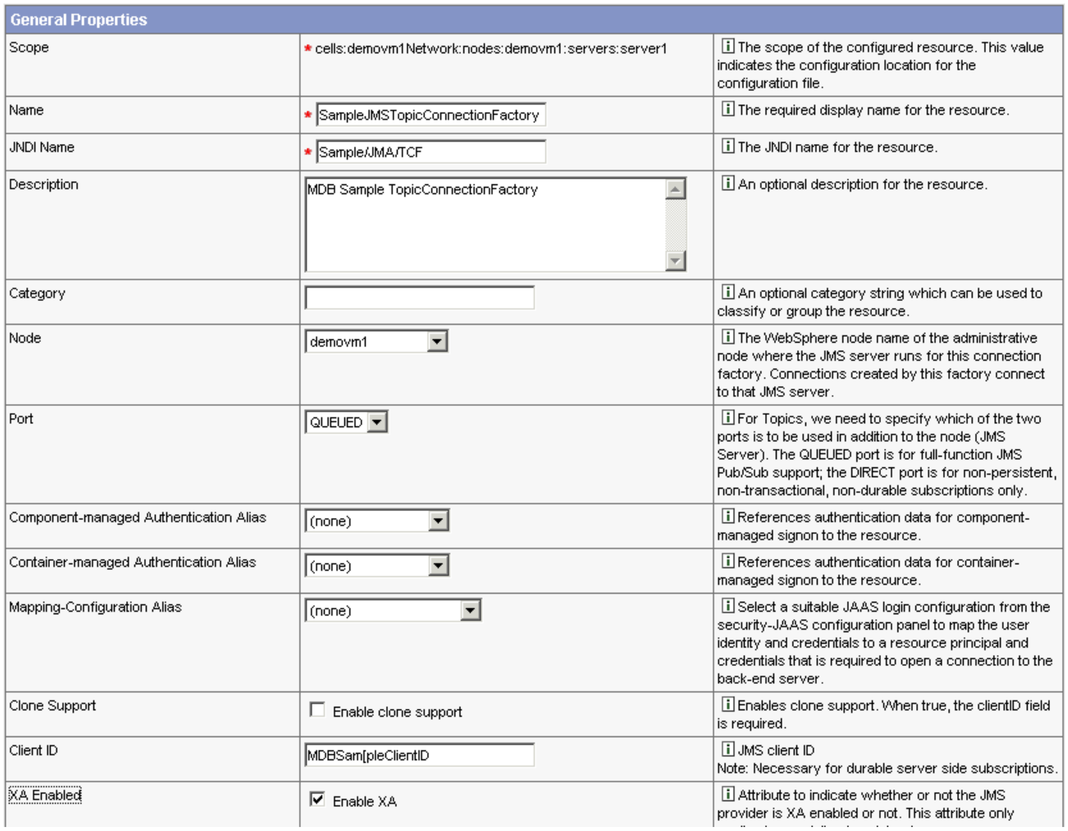
WebSphere Configuration Problems

Figure 11-21 Topic connection factory objects

   In this example, we have one WebSphere topic connection factory object, SampleJMSTopicConnectionFactory. This topic connection factory object is used to connect to a WebSphere JMS provider in the cell.

4. To create a new topic connection factory object, click **New** in the content pane. To change the properties of an existing topic connection factory, click one of the connection factories displayed.

   Figure 11-22 on page 515, shows the properties for the SampleJMSTopicConnectionFactory topic connection factory object.



General Properties

Scope	* cells:demovm1Network:nodes:demovm1:servers:server1	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* SampleJMSTopicConnectionFactory	The required display name for the resource.
JNDI Name	* Sample/JMA/TCF	The JNDI name for the resource.
Description	MDB Sample TopicConnectionFactory	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Node	demovm1	The WebSphere node name of the administrative node where the JMS server runs for this connection factory. Connections created by this factory connect to that JMS server.
Port	QUEUED	For Topics, we need to specify which of the two ports is to be used in addition to the node (JMS Server). The QUEUED port is for full-function JMS Pub/Sub support; the DIRECT port is for non-persistent, non-transactional, non-durable subscriptions only.
Component-managed Authentication Alias	(none)	References authentication data for component-managed signon to the resource.
Container-managed Authentication Alias	(none)	References authentication data for container-managed signon to the resource.
Mapping-Configuration Alias	(none)	Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and credentials to a resource principal and credentials that is required to open a connection to the back-end server.
Clone Support	☐ Enable clone support	Enables clone support. When true, the clientID field is required.
Client ID	MDBSam[pleClientID	JMS client ID Note: Necessary for durable server side subscriptions.
XA Enabled	✓ Enable XA	Attribute to indicate whether or not the JMS provider is XA enabled or not. This attribute only

Figure 11-22 Topic connection factory properties

The configuration properties to note are:

– **JNDI name**: Used to bind the topic connection factory into the application server’s name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

– **Node:** The node name of the administrative node where the JMS provider for this topic connection factory runs. Connections created by this factory connect to that publish/subscribe JMS provider.

> **Note:** The node name is the WebSphere logical node name, not the host name of the machine.

– **Component-managed/Container-managed authentication alias:** Used to reference the J2C authentication data entry (in the Additional Properties table) that defines the user ID/password to be used to authenticate connection to a JMS provider.

– **Port:** Used to connect to the publish/subscribe JMS provider. The two possible values are:

  i. QUEUED

     This is the port used by the JMS provider queue manager. If the topic connection factory object is configured with Port set to `QUEUED`, the JMS publish/subscribe provider is accessed via the WebSphere JMS provider queue manager. The queued port is the port used for full-function JMS publish/subscribe support.

  ii. DIRECT

     The direct port is the publish/subscribe JMS provider listener port used for non-persistent, non-transactional, non-durable subscriptions. If the topic connection factory object is configured with Port set to `DIRECT`, the JMS publish/subscribe provider is accessed directly on this port via TCP/IP.

     Note: Message-driven beans cannot use the DIRECT listener port for publish/subscribe support. Therefore, any topic connection factory object configured with Port set to `DIRECT` cannot be used with message-driven beans.

  The TCP/IP listener ports to access the publish/subscribe JMS provider are defined in the serverindex.xml file for the node in which the WebSphere JMS provider is installed, under the servername=”jmsserver” stanza.

– The JMS client identifier used for connections to the publish/subscribe JMS provider.

– Clone Support enables the JMS CloneSupport feature. This feature is for durable topic subscriptions. It enables sharing of durable subscriptions across WebSphere cluster members. This means you can have the same durable subscription on separate cluster members such that a single publication will only be delivered to one of them; hence a stream of publications get distributed across the servers.

Other configurable properties include:

- Connection pool settings
- Session pool settings
- J2C authentication data entries

Note: JMS pooling is managed by the J2C Connection Manager.

5. Click **OK** when you have finished configuring the topic connection factory object and save the configuration.

6. To have the changed configuration take effect, stop then restart the application servers using the topic connection factory object.

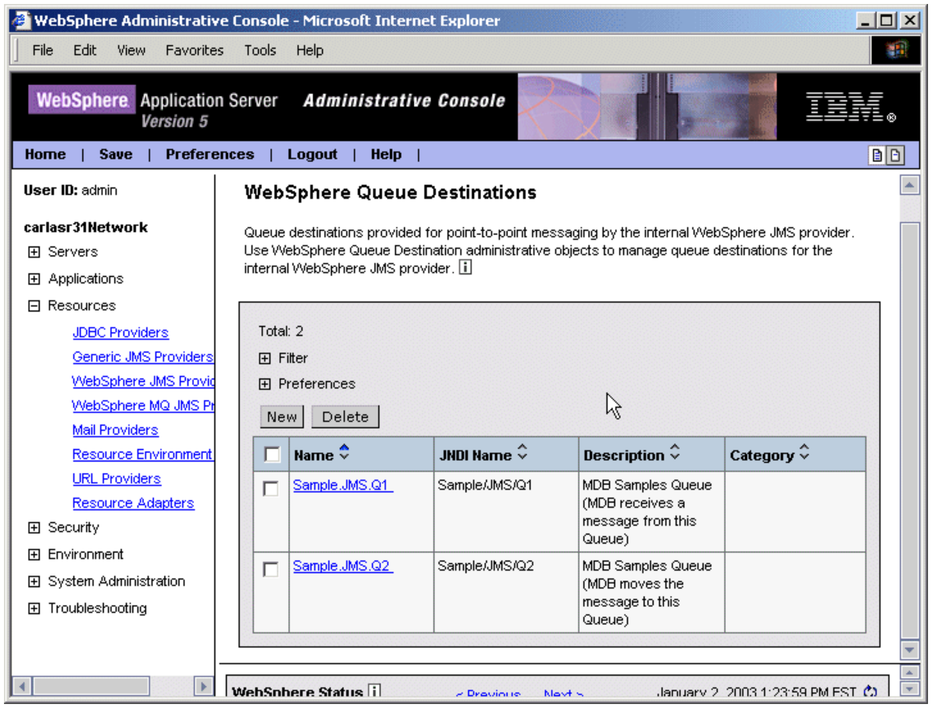
Configuring queue destinations

A queue destination object is a WebSphere administrative object used to represent a JMS messaging system queue. Connections to the queue are made via a queue connection factory object that connects to the embedded WebSphere JMS provider.

To configure the properties of a queue destination object, complete these steps:

1. In the navigation tree, expand **Resources -> WebSphere JMS Providers**.
2. Select **Scope** and click **Apply**. For a discussion on the scope setting, see "Configuring a queue connection factory" on page 510.
3. Click **WebSphere Queue Destinations** in the Additional Properties table.

   This displays WebSphere queue destination objects available to application servers under the selected scope. These objects represent messaging queues hosted by an WebSphere JMS provider in the cell. This is shown in Figure 11-23 on page 518.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |

User ID: admin

carlasr31Network

- Servers
- Applications
- Resources
  - JDBC Providers
  - Generic JMS Providers
  - WebSphere JMS Provid
  - WebSphere MQ JMS Pr
  - Mail Providers
  - Resource Environment
  - URL Providers
  - Resource Adapters
- Security
- Environment
- System Administration
- Troubleshooting

WebSphere Queue Destinations

Queue destinations provided for point-to-point messaging by the internal WebSphere JMS provider. Use WebSphere Queue Destination administrative objects to manage queue destinations for the internal WebSphere JMS provider.

Total: 2

- Filter
- Preferences

New Delete

	Name	JNDI Name	Description	Category
☐	Sample.JMS.Q1	Sample/JMS/Q1	MDB Samples Queue (MDB receives a message from this Queue)	
☐	Sample.JMS.Q2	Sample/JMS/Q2	MDB Samples Queue (MDB moves the message to this Queue)	

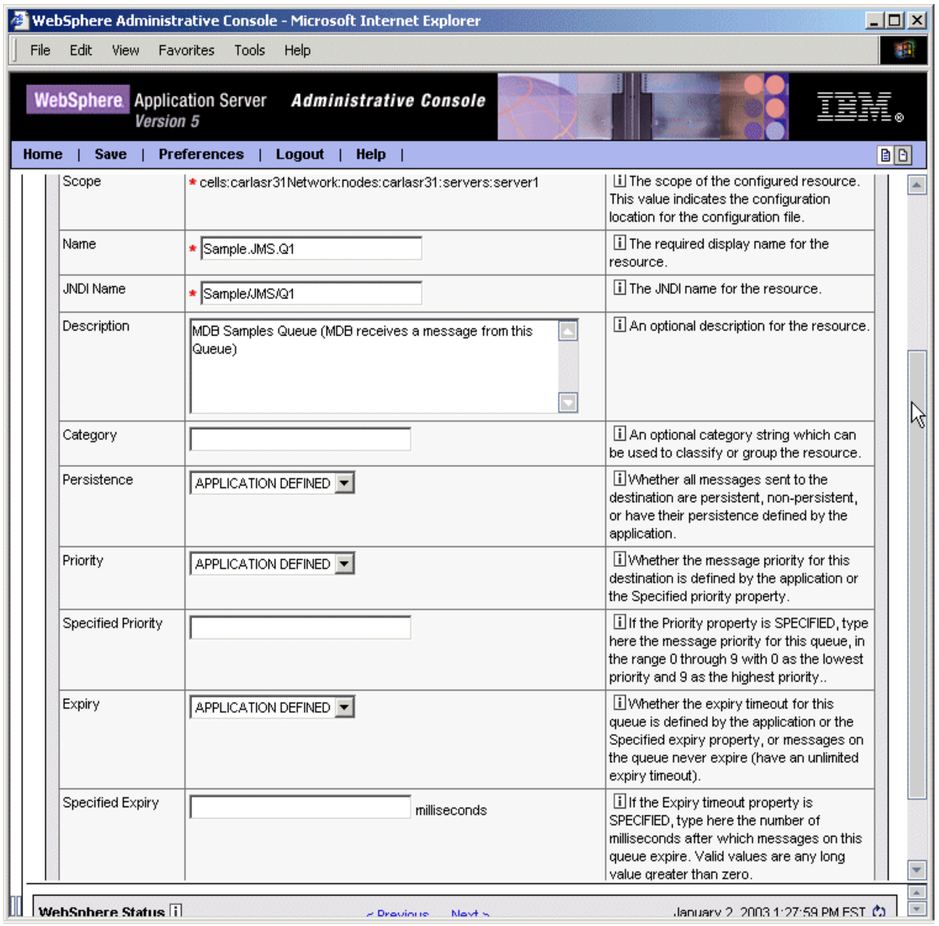
WebSphere Status < Previous Next > January 2, 2003 1:23:59 PM EST

Figure 11-23 Queue destination administered objects at the selected scope

In this example we have two WebSphere queue destination objects, Sample.JMS.Q1 and Sample.JMS.Q2.

4. To create a new queue destination object, click **New**, or to change the properties of an existing queue destination object, click one of the queue destination objects displayed.

   Figure 11-24 on page 519 shows the properties for the Sample.JMS.Q1 queue destination object.


WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

Scope	* cells:carlasr31Network:nodes:carlasr31:servers:server1	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* Sample.JMS.Q1	The required display name for the resource.
JNDI Name	* Sample/JMS/Q1	The JNDI name for the resource.
Description	MDB Samples Queue (MDB receives a message from this Queue)	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	Whether the message priority for this destination is defined by the application or the Specified priority property.
Specified Priority		If the Priority property is SPECIFIED, type here the message priority for this queue, in the range 0 through 9 with 0 as the lowest priority and 9 as the highest priority..
Expiry	APPLICATION DEFINED	Whether the expiry timeout for this queue is defined by the application or the Specified expiry property, or messages on the queue never expire (have an unlimited expiry timeout).
Specified Expiry	milliseconds	If the Expiry timeout property is SPECIFIED, type here the number of milliseconds after which messages on this queue expire. Valid values are any long value greater than zero.

WebSphere Status < Previous Next > January 2, 2003 1:27:59 PM EST

Figure 11-24 Queue destination object properties

The configuration properties to note are:

– **Name:** Used to refer to the queue for administrative purposes. It is this name that must be added to the JMS server queue names list in order to make the queue available to applications (see Figure 11-18 on page 505). Adding the queue name to the JMS server creates the messaging queue for the WebSphere JMS provider, but you won't be able to see it or administer it.

– **JNDI name:** Used to bind the queue into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

   – **Persistence:** Specifies whether all messages sent to the destination are PERSISTENT, NON PERSISTENT, or have their persistence defined by the application, APPLICATION DEFINED. The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that instructs the JMS provider whether to store the message in case of failure.

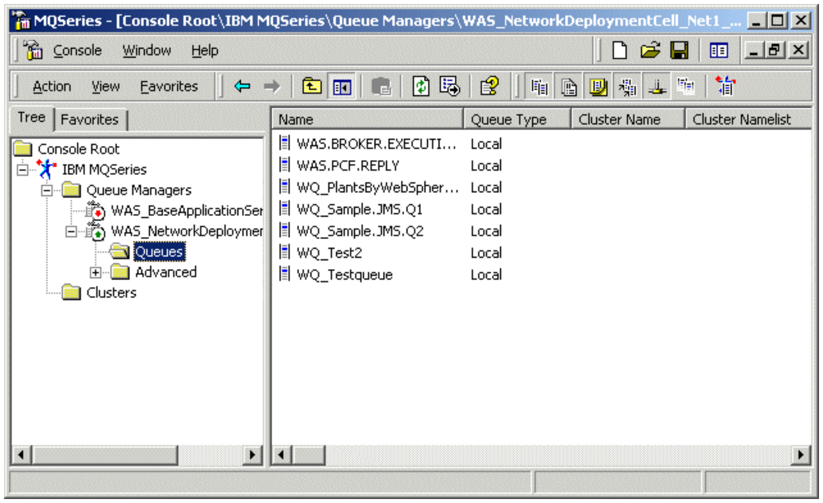
5. Click **OK** when finished configuring the queue destination object and save the configuration.
6. To have the changed configuration take effect, stop then restart the application servers using the resource.

Note: If you install the full version of WebSphere MQ after installing the embedded WebSphere JMS provider, existing JMS resource definitions for the WebSphere JMS provider will continue to work with WebSphere MQ as the JMS provider, so you do not need to redefine those JMS resources.

Figure 11-25, for example, shows the WebSphere MQ Explorer after installing WebSphere MQ over the embedded WebSphere JMS provider. We can see the queue managers previously created when installing the WebSphere JMS provider, such as WAS_<cellname>_Net1_AS, and the queues defined to the JMS server process, such as Sample.JMS.Q1.

The queues actually created in the internal queue manager have the name WQ_<JMS server queuename>.

With WebSphere MQ Explorer, we can now manage those queues. This was not possible with the embedded WebSphere JMS provider.



Name	Queue Type	Cluster Name	Cluster Namelist
WAS.BROKER.EXECUTI...	Local		
WAS.PCF.REPLY	Local		
WQ_PlantsByWebSpher...	Local		
WQ_Sample.JMS.Q1	Local		
WQ_Sample.JMS.Q2	Local		
WQ_Test2	Local		
WQ_Testqueue	Local		

Figure 11-25 Installing WebSphere MQ over the WebSphere JMS provider

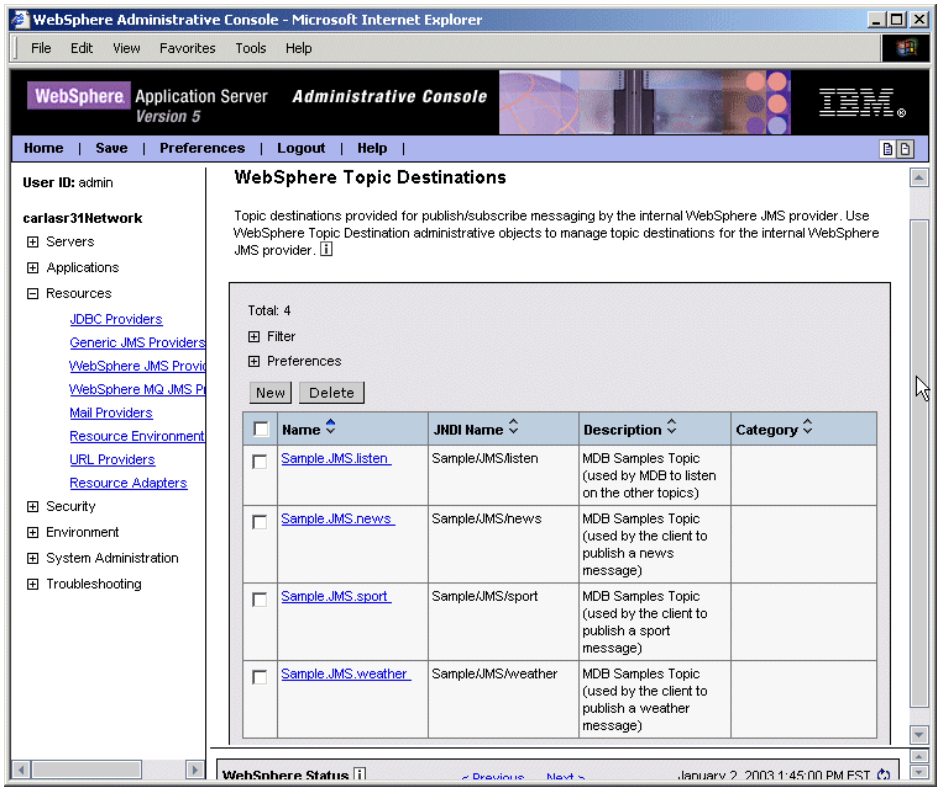
Configuring topic destinations

A topic destination is used to configure the properties of a WebSphere administrative object that represents a JMS topic in the messaging system. Connections to the topic are created by a topic connection factory that connects to the publish/subscribe provider installed with the embedded WebSphere JMS provider.

To configure the properties of a topic destination administrative object, which represents a topic served by the publish/subscribe JMS provider that gets installed with the WebSphere JMS provider, complete these steps:

1. In the navigation tree, expand **Resources -> WebSphere JMS Providers**.
2. Select **Scope** and click **Apply**. For a discussion on the scope setting, see “Configuring a queue connection factory” on page 510.
3. Click **WebSphere Topic Destinations** in the Additional Properties table.

   This displays WebSphere topic destination objects available to application servers under the selected scope. These objects represent actual topics served by an embedded publish/subscribe JMS provider on the cell. This is shown in Figure 11-26 on page 522.



WebSphere Topic Destinations

Topic destinations provided for publish/subscribe messaging by the internal WebSphere JMS provider. Use WebSphere Topic Destination administrative objects to manage topic destinations for the internal WebSphere JMS provider.

Total: 4

Filter

Preferences

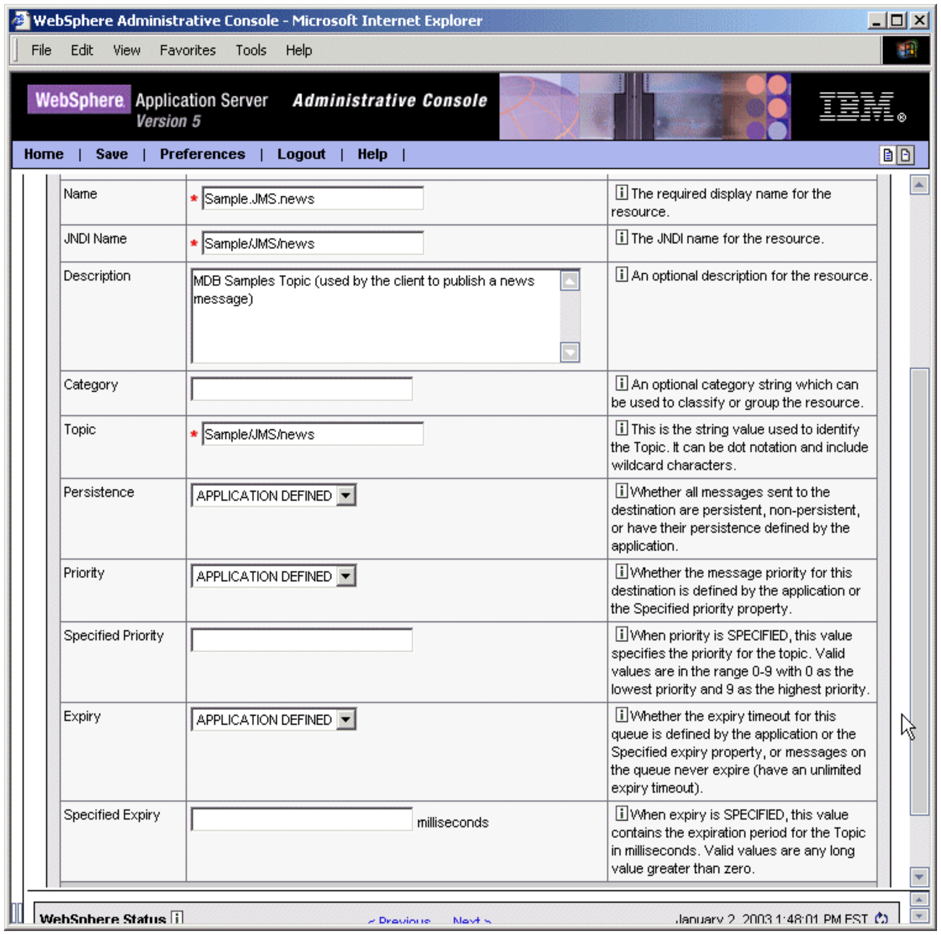
	Name	JNDI Name	Description	Category
☐	Sample.JMS.listen	Sample/JMS/listen	MDB Samples Topic (used by MDB to listen on the other topics)	
☐	Sample.JMS.news	Sample/JMS/news	MDB Samples Topic (used by the client to publish a news message)	
☐	Sample.JMS.sport	Sample/JMS/sport	MDB Samples Topic (used by the client to publish a sport message)	
☐	Sample.JMS.weather	Sample/JMS/weather	MDB Samples Topic (used by the client to publish a weather message)	

Figure 11-26 Topic destination administered objects at the selected scope

In this example, we have four WebSphere topic destination objects defined, representing topics served by the publish/subscribe provider installed with the WebSphere JMS provider.

4. To create a new topic destination object, click **New** in the content pane, as shown in Figure 11-26, or to change the properties of an existing topic destination object, click one of the topic destination objects displayed.

   Figure 11-27 on page 523, shows the properties for the Sample.JMS.news topic destination object.



WebSphere Administrative Console - Microsoft Internet Explorer

Name	* Sample.JMS.news	The required display name for the resource.
JNDI Name	* Sample/JMS/news	The JNDI name for the resource.
Description	MDB Samples Topic (used by the client to publish a news message)	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Topic	* Sample/JMS/news	This is the string value used to identify the Topic. It can be dot notation and include wildcard characters.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	Whether the message priority for this destination is defined by the application or the Specified priority property.
Specified Priority		When priority is SPECIFIED, this value specifies the priority for the topic. Valid values are in the range 0-9 with 0 as the lowest priority and 9 as the highest priority.
Expiry	APPLICATION DEFINED	Whether the expiry timeout for this queue is defined by the application or the Specified expiry property, or messages on the queue never expire (have an unlimited expiry timeout).
Specified Expiry	milliseconds	When expiry is SPECIFIED, this value contains the expiration period for the Topic in milliseconds. Valid values are any long value greater than zero.

WebSphere Status < Previous Next > January 2, 2003 1:48:01 PM EST

Figure 11-27 Topic destination object properties

The configuration properties to note are:

– **JNDI name:** Binds the topic destination into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

– **Topic:** Specifies the name of the topic defined to the publish/subscribe provider.

Note: The message topic for the publish/subscribe JMS provider gets created dynamically when the application invokes the administered object.

5. Click **OK** when finished configuring the topic destination object and save the configuration.
6. To have the changed configuration take effect, stop then restart the application servers using the resource.

11.8.6 Configuring resources for WebSphere MQ JMS provider

If you are using the WebSphere MQ JMS provider or a publish/subscribe provider installed with WebSphere MQ, to support enterprise applications using JMS, you can still use the administrative console to configure connection factories and destination objects into the WebSphere name space.

The administrative objects that you configure in WebSphere must match the JMS messaging resources created in WebSphere MQ or the publish/subscribe provider:

- Configure queue connection factory.
- Configure topic connection factory.
- Configure queue destination.
- Configure topic destination.

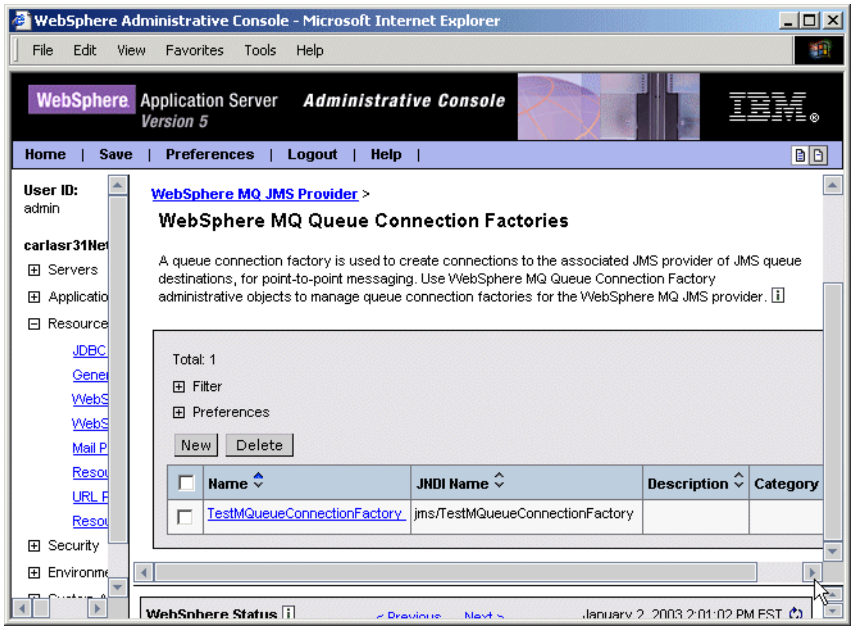
Note: WebSphere MQ resources do not get created automatically for you, as was the case with the embedded WebSphere JMS provider. You need to create those messaging resources in WebSphere MQ, using WebSphere MQ Explorer wizard for example.

Configuring queue connection factories

Complete the following steps to configure properties for a queue connection factory object that will be used to connect to a WebSphere MQ JMS provider:

1. In the navigation tree, expand **Resources -> WebSphere MQ JMS Providers**.
2. Select **Scope** and click **Apply**. For a discussion on the scope setting, see “Configuring a queue connection factory” on page 510.
3. Click **WebSphere MQ Queue Connection Factories** in the Additional Properties table.

   This displays WebSphere MQ queue connection factory administered objects available to application servers under the selected scope, and which are used to connect to a WebSphere MQ JMS provider. This is shown in Figure 11-28 on page 525.


WebSphere MQ JMS Provider >

WebSphere MQ Queue Connection Factories

A queue connection factory is used to create connections to the associated JMS provider of JMS queue destinations, for point-to-point messaging. Use WebSphere MQ Queue Connection Factory administrative objects to manage queue connection factories for the WebSphere MQ JMS provider.

Total: 1

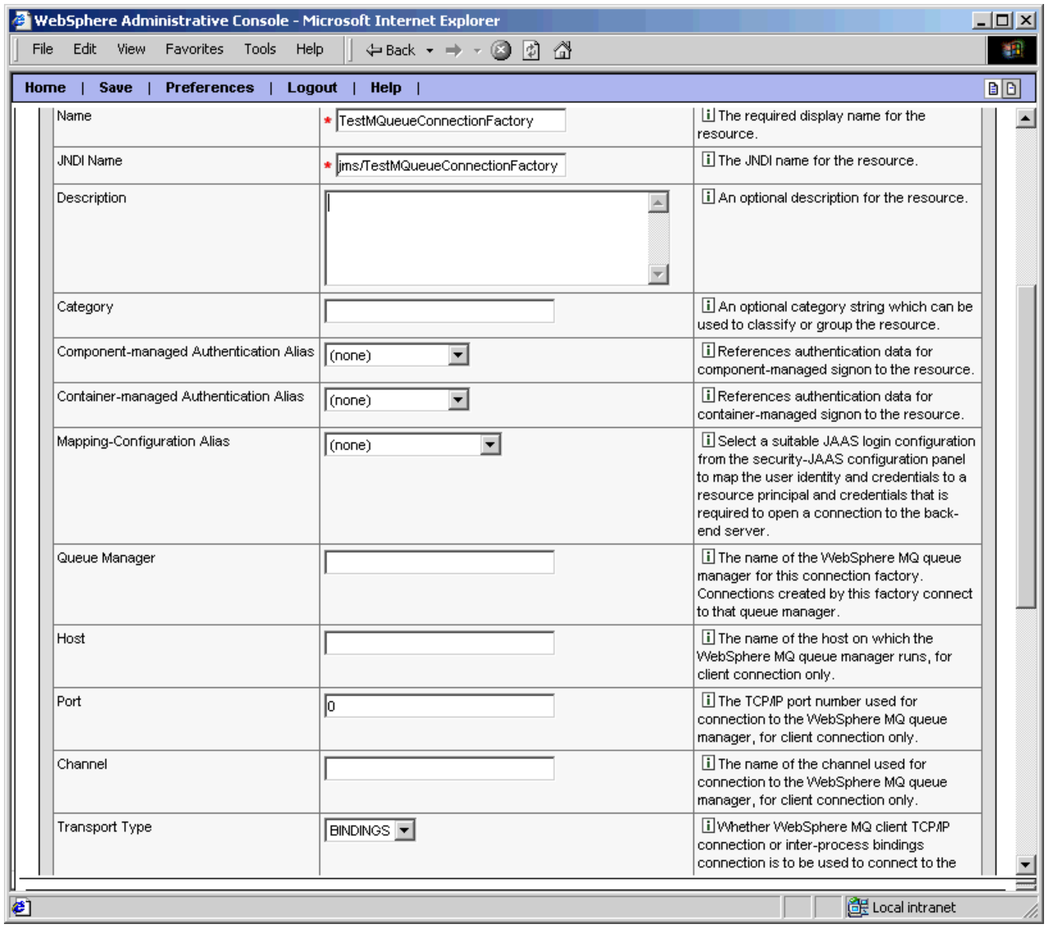
	Name	JNDI Name	Description	Category
☐	TestMQueueConnectionFactory	jms/TestMQueueConnectionFactory		

Figure 11-28 MQQueue connection factory administered objects

In this example, we have one WebSphere MQ queue connection factory object, TestMQueueConnectionFactory, defined. This connection factory object has all the necessary properties to connect to a full WebSphere MQ JMS provider.

4. To create a new queue connection factory object, click **New**, or to change the properties of an existing queue connection factory, click one of the connection factories displayed.

   Figure 11-29 on page 526 partially shows the properties available to specify for a new queue connection factory object.


WebSphere Administrative Console - Microsoft Internet Explorer

Home | Save | Preferences | Logout | Help |

Name	* TestMQueueConnectionFactory	The required display name for the resource.
JNDI Name	* jms/TestMQueueConnectionFactory	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Component-managed Authentication Alias	(none)	References authentication data for component-managed signon to the resource.
Container-managed Authentication Alias	(none)	References authentication data for container-managed signon to the resource.
Mapping-Configuration Alias	(none)	Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and credentials to a resource principal and credentials that is required to open a connection to the back-end server.
Queue Manager		The name of the WebSphere MQ queue manager for this connection factory. Connections created by this factory connect to that queue manager.
Host		The name of the host on which the WebSphere MQ queue manager runs, for client connection only.
Port	0	The TCP/IP port number used for connection to the WebSphere MQ queue manager, for client connection only.
Channel		The name of the channel used for connection to the WebSphere MQ queue manager, for client connection only.
Transport Type	BINDINGS	Whether WebSphere MQ client TCP/IP connection or inter-process bindings connection is to be used to connect to the

Local intranet

Figure 11-29 WebSphere MQ queue connection factory properties

- **Name (required):** Name used for object administration purposes.
- **JNDI name (required):** Used to bind the connection factory into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.
- **Component-managed/container-managed authentication alias:** Used to reference the J2C authentication data entry (in the Additional Properties table) that defines the user ID/password to be used to authenticate a connection to a JMS provider.
- **Mapping-Configuration Alias:** Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and

credentials to a resource principal and credentials required to open a connection to the back-end server.

– **Queue Manager:** defines the name of the WebSphere MQ provider queue manager that we will be connecting to. Connections created by this factory connect to that queue manager.

Note: The queue manager is created and managed from WebSphere MQ, (with the WebSphere MQ Explorer wizard for example). For a full WebSphere MQ provider, you can have any number of queue managers for that provider, and with any name you want. For the embedded WebSphere JMS provider, this was not possible. You only have one queue manager for the embedded provider.

– **Host (required for client transport):** The name of the host on which the WebSphere MQ queue manager runs, for client connection only.

– **Port (required for client transport):** The TCP/IP port number used for connection to the WebSphere MQ queue manager, for client connection only. This is the port used by the queue manager listener.

– **Channel:** The name of the channel used for connection to the WebSphere MQ queue manager, for client connection only.

– **Transport Type (required):** The transport type used for communication between the JMS application and the WebSphere MQ provider queue manager. The possible values are BINDINGS (WebSphere MQ bindings) and CLIENT (TCP/IP). The default is BINDINGS.

  BINDINGS requires the JMS application and the WebSphere MQ server be located on the same machine. CLIENT permits the queue manager to be on a different machine from the application.

  If we select bindings as the transport type, the connection factory specifies connection properties to communicate with a local WebSphere MQ provider queue manager.

  Note: For detailed information about WebSphere MQ client and bind modes, see the *MQSeries System Administration* document, SC33-1873-02.

– **Model Queue Definition:** The name of the model queue definition that can be used by the queue manager to create temporary queues if a requested queue does not already exist.

– **Client ID:** The JMS client identifier used for connections to the WebSphere MQ queue manager.

– **CCSID:** The coded character set identifier used to connect to the WebSphere MQ provider queue manager. This coded character set identifier must be one of the CCSIDs supported by WebSphere MQ.

Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

– **Message Retention:** Check this box if you want unwanted messages to be left on the queue. Otherwise, unwanted messages will be dealt with according to their disposition options.

– **XA Enabled:** Specifies whether the connection factory is for XA or non-XA coordination of messages and controls if the application server uses XA QCF/TCF. This attribute only applies to specialized models of JMSConnectionFactory.

Other properties configurable in the Additional Properties table are connection pool settings and session pool settings.

Note: JMS connection and session resources are managed by the J2C Connection Manager.

5. Click **OK** when finished and save the configuration.
6. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configuring topic connection factories

A topic connection factory object is used to create JMS connections to the JMS publish/subscribe provider for publish/subscribe messaging to topic destinations. In order to use this support, you will need to have a publish/subscribe provider installed. In this section we assume that one of these packages has been installed for publish/subscribe support.

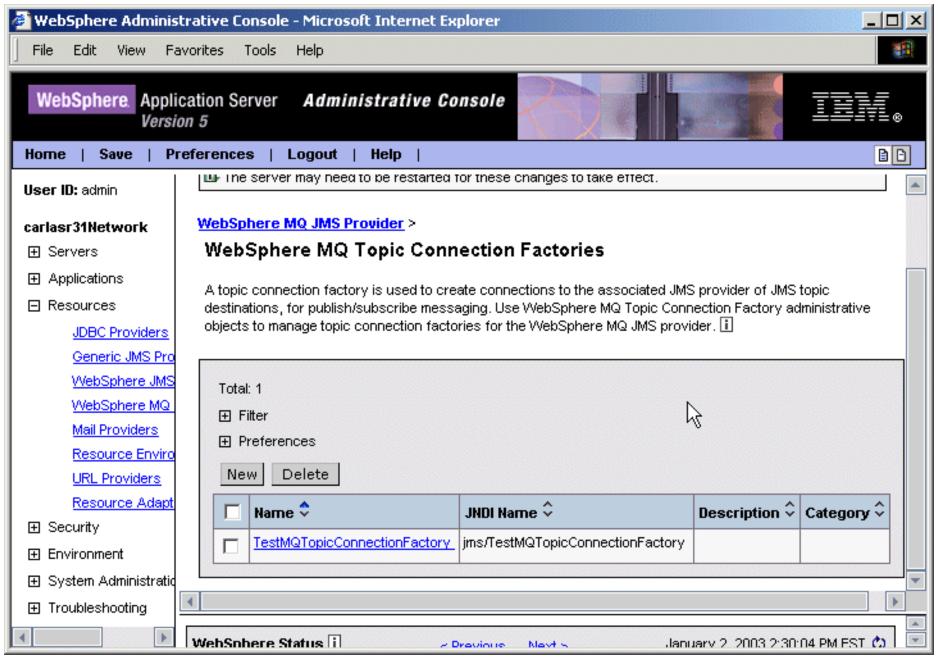
Publish/subscribe providers: The supported providers include:

- MQ Publish and Subscribe
- WebSphere MQ Integrator
- WebSphere MQ Publish/Subscribe MA0C SupportPac
- WebSphere MQ Event Broker.

To configure the properties of a topic connection factory object, which will be used to connect to the publish/subscribe provider installed with WebSphere MQ, complete these steps:

1. In the navigation tree, expand **Resources -> WebSphere MQ JMS Providers**.
2. Select **Scope** and click **Apply**. For a discussion on the scope setting, see “Configuring a queue connection factory” on page 510.
3. Click **WebSphere MQ Topic Connection Factories** in the Additional Properties table.

   This displays WebSphere MQ topic connection factory administered objects available to application servers under the selected scope, and which are used to connect to a publish/subscribe provider installed with WebSphere MQ. This is shown in Figure 11-30.



The server may need to be restarted for these changes to take effect.

WebSphere MQ JMS Provider >

WebSphere MQ Topic Connection Factories

A topic connection factory is used to create connections to the associated JMS provider of JMS topic destinations, for publish/subscribe messaging. Use WebSphere MQ Topic Connection Factory administrative objects to manage topic connection factories for the WebSphere MQ JMS provider.

Total: 1

Filter

Preferences

New Delete

	Name	JNDI Name	Description	Category
☐	TestMQTopicConnectionFactory	jms/TestMQTopicConnectionFactory		

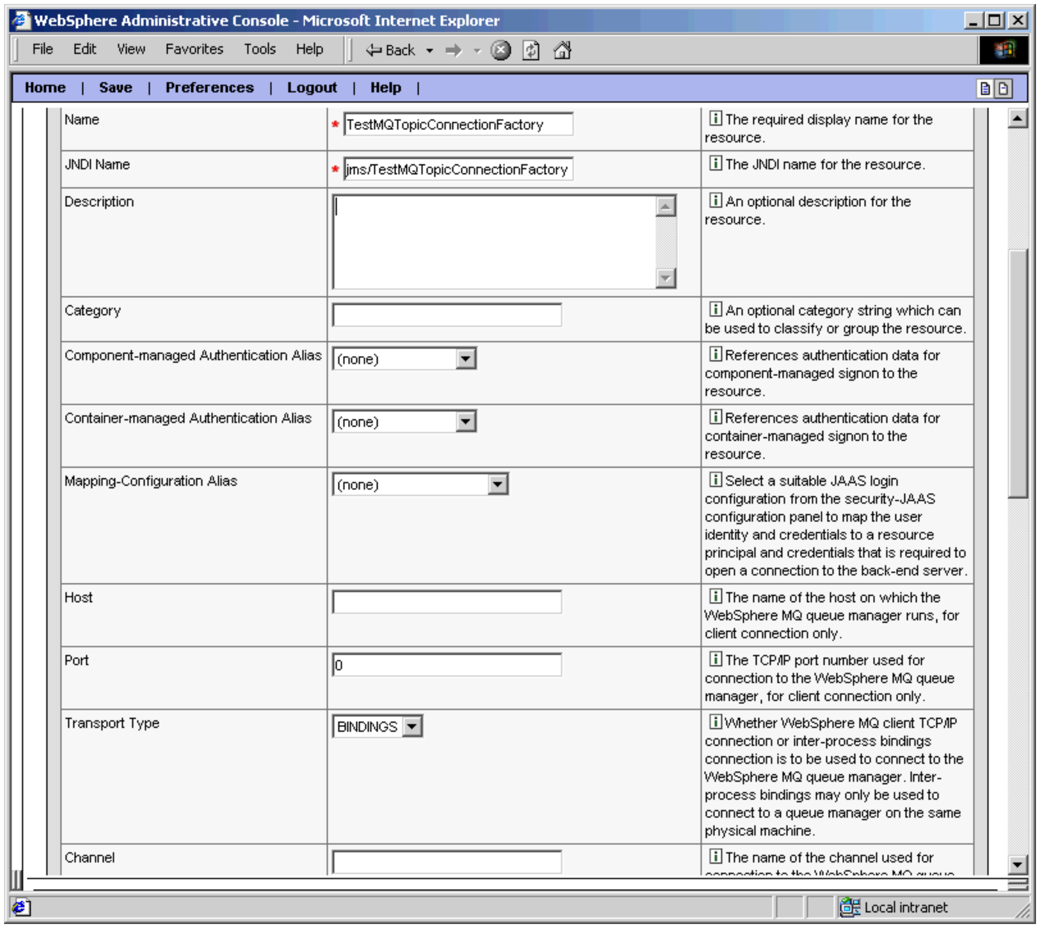
WebSphere Status < Previous Next > January 2, 2003 2:30:04 PM EST

Figure 11-30 WebSphere MQ topic connection factory objects

   In this example we have one WebSphere MQ topic connection factory object defined. This topic connection factory object is used to connect to a publish/subscribe provider installed with WebSphere MQ.

4. To create a new topic connection factory object, click **New,** or to change the properties of an existing topic connection factory, click one of the connection

factories displayed. Figure 11-31 shows the properties for available to define a topic connection factory object.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Home | Save | Preferences | Logout | Help |
Local intranet

Name	* TestMQTopicConnectionFactory	The required display name for the resource.
JNDI Name	* jms/TestMQTopicConnectionFactory	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Component-managed Authentication Alias	(none)	References authentication data for component-managed signon to the resource.
Container-managed Authentication Alias	(none)	References authentication data for container-managed signon to the resource.
Mapping-Configuration Alias	(none)	Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and credentials to a resource principal and credentials that is required to open a connection to the back-end server.
Host		The name of the host on which the WebSphere MQ queue manager runs, for client connection only.
Port	0	The TCP/IP port number used for connection to the WebSphere MQ queue manager, for client connection only.
Transport Type	BINDINGS	Whether WebSphere MQ client TCP/IP connection or inter-process bindings connection is to be used to connect to the WebSphere MQ queue manager. Inter-process bindings may only be used to connect to a queue manager on the same physical machine.
Channel		The name of the channel used for

Figure 11-31 WebSphere MQ topic connection factory properties

The configuration properties to note are:

– **JNDI name:** The JNDI name used to bind the connection factory into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

– **Component-managed/Container-managed authentication alias:** Used to reference the J2C authentication data entry (in the Additional Properties table) that defines the user ID/password to be used to authenticate connection to a JMS provider.

– **Mapping-Configuration Alias:** Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and credentials to a resource principal and credentials required to open a connection to the back-end server.

– **Host:** The name of the host on which the WebSphere MQ queue manager that hosts the broker (publish/subscribe provider) runs, for client connection only.

– **Port:** The TCP/IP port number used for connection to the WebSphere MQ queue manager that hosts the broker, for client connection only. This is the port used by the queue manager listener.

– **Transport Type:** The transport type used for communication between the JMS application and the WebSphere MQ provider queue manager that hosts the broker. The possible values are BINDINGS (WebSphere MQ bindings) and CLIENT (TCP/IP). The default is BINDINGS.

  BINDINGS requires the JMS application and the WebSphere MQ server to be located on the same machine. CLIENT permits the queue manager to be on a different machine from the application.

  Note: For detailed information on WebSphere MQ client and bind modes, see the *MQSeries System Administration* document, SC33-1873-02.

– **Channel:** The name of the channel used for connection to the WebSphere MQ queue manager that hosts the broker, for client connection only.

– **Queue manager:** The name of the WebSphere MQ provider queue for this connection factory. Connections created by this factory connect to that queue manager.

– **Broker Control Queue:** The name of the broker's control queue to which all command messages (except `Publish` and `Delete Publication` command messages) are sent. Publisher and subscriber applications and other brokers send command messages to this queue.

  For example, a publisher that wishes to register with the broker would send a `Register Publisher` command message to the broker's control queue.

– **Broker Queue Manager:** The name of the WebSphere MQ queue manager that provides the publish/subscribe message broker.

– **Broker Publication Queue:** The name of the broker's input queue used to submit publications, for example SYSTEM.BROKER.DEFAULT.STREAM. The Broker can have multiple publication queues. You can set up a topic connection factory for each one of them.

– **Broker Subscription Queue:** The name of the broker's queue from which non-durable subscription messages are retrieved. A subscriber can have an exclusive queue assigned to it from which it retrieves all its messages, or it can use a shared queue, such as SYSTEM.JMS.ND.SUBSCRIPTION.QUEUE, from which it and other subscribers retrieve their messages.

– **Broker CC Subscription Queue**: The name of the broker's queue from which non-durable subscription messages are retrieved for a ConnectionConsumer.

Note: The above queues are WebSphere MQ queues managed by the broker queue manager.

– **Broker Version:** Whether the message broker is provided by the WebSphere MQ MA0C Supportpac or newer versions (MQ Integrator and MQ Publish and Subscribe).

– **Model Queue Definition:** The name of the model queue definition that the broker can use to create dynamic queues to receive publications for streams other than the default stream. This is only used if the stream queue does not already exist. If this model queue definition does not exist, all stream queues must be defined by the administrator.

– **Clone support:** Select this check box to enable WebSphere MQ clone support to allow the same durable subscription across topic clones.

– **Client ID:** The JMS client identifier used for connections to the WebSphere MQ queue manager hosting the broker.

– **CCSID:** The coded character set identifier used to connect to the WebSphere MQ provider queue manager hosting the broker. This coded character set identifier must be one of the CCSIDs supported by WebSphere MQ.

Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

– **XA Enabled:** Specifies whether the connection factory is for XA or non-XA coordination of messages and controls whether the application server uses XA QCF/TCF.

Other configurable properties are connection pool settings, session pool settings, and J2C authentication data entries.

Note: JMS connection and session resources are managed by the J2C Connection Manager.

5. Click **OK** when you have finished and save the configuration.
6. To have the changed configuration take effect, stop then restart the application servers using the topic connection factory object.

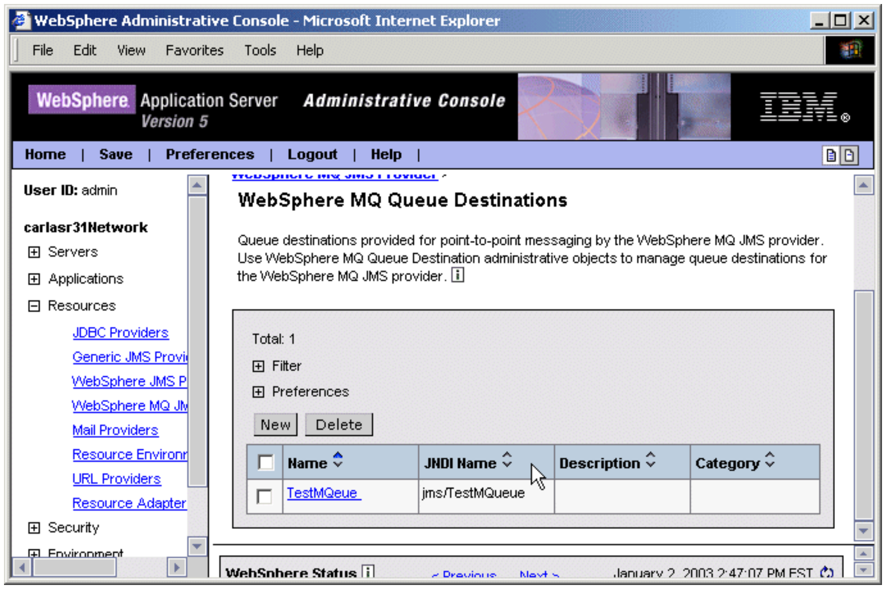
Configuring queue destinations

Complete the following steps to configure the properties of a queue destination object, that represents a queue hosted by the WebSphere MQ JMS provider:

1. In the navigation tree, expand **Resources -> WebSphere MQ JMS Providers**.
2. Select **Scope** and click **Apply**.

   For a discussion on the scope setting, see “Configuring a queue connection factory” on page 510.
3. Click **WebSphere MQ Queue Destinations** in the Additional Properties table.

   This displays WebSphere MQ queue destination objects available to application servers under the selected scope. These objects represent messaging queues hosted by a WebSphere MQ JMS provider. This is shown in Figure 11-32 on page 534.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Network
Servers
Applications
Resources
JDBC Providers
Generic JMS Provi
WebSphere JMS P
WebSphere MQ JM
Mail Providers
Resource Environ
URL Providers
Resource Adapter
Security
Environment
WebSphere Status
< Previous
Next >
January 2, 2003 2:47:07 PM EST

WebSphere MQ Queue Destinations

Queue destinations provided for point-to-point messaging by the WebSphere MQ JMS provider. Use WebSphere MQ Queue Destination administrative objects to manage queue destinations for the WebSphere MQ JMS provider.

Total: 1

Filter

Preferences

New Delete

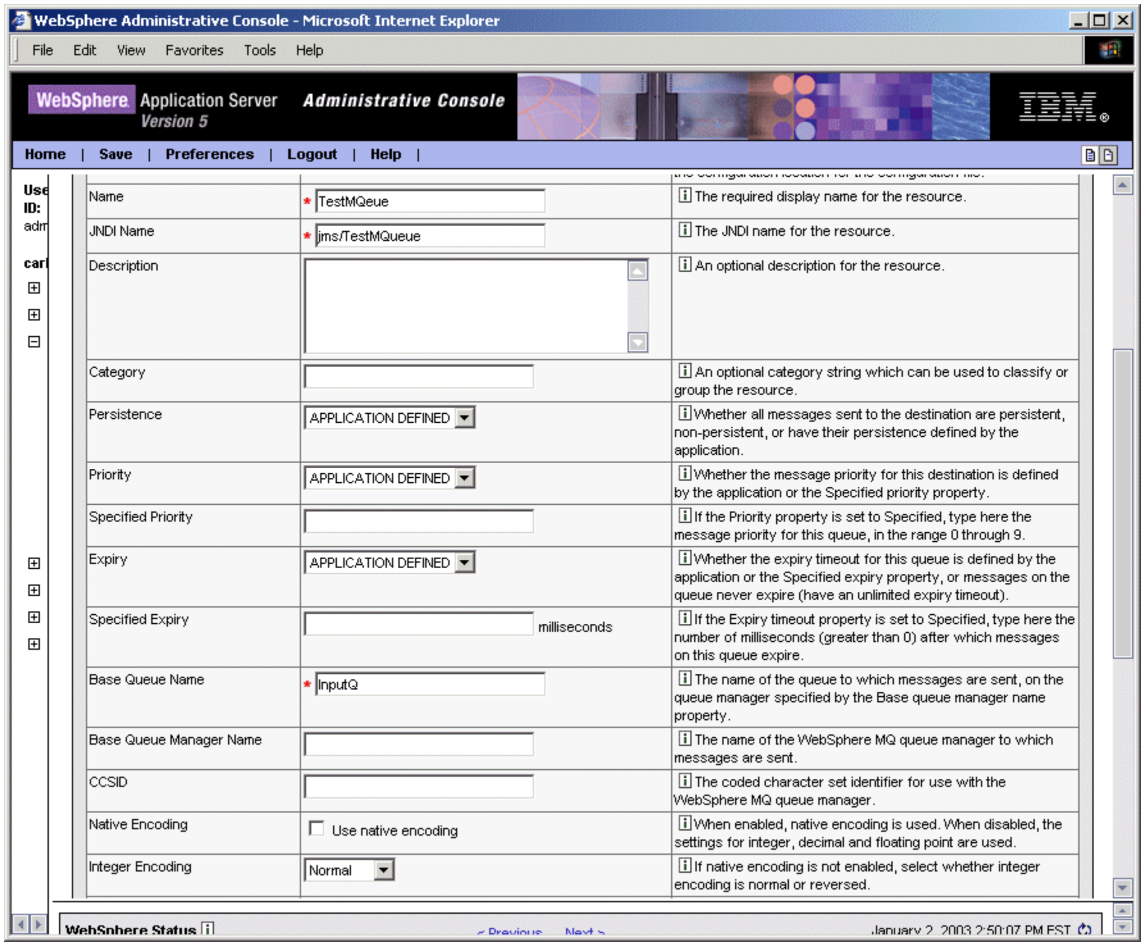
	Name	JNDI Name	Description	Category
☐	TestMQeue	jms/TestMQueue		

Figure 11-32 WebSphere MQ queue destination administered objects

In this example we have one WebSphere MQ queue destination object, TestMQueue, defined. This queue destination object represents a queue hosted by a WebSphere MQ JMS provider.

4. To create a new queue destination object, click **New** in the content pane, or to change the properties of an existing queue destination object, click one of the queue destination objects displayed.

   Figure 11-33 on page 535 shows the destination queue configuration page.



WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Home | Save | Preferences | Logout | Help |

Name	* TestMQeue	The required display name for the resource.
JNDI Name	* jms/TestMQueue	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	Whether the message priority for this destination is defined by the application or the Specified priority property.
Specified Priority		If the Priority property is set to Specified, type here the message priority for this queue, in the range 0 through 9.
Expiry	APPLICATION DEFINED	Whether the expiry timeout for this queue is defined by the application or the Specified expiry property, or messages on the queue never expire (have an unlimited expiry timeout).
Specified Expiry	milliseconds	If the Expiry timeout property is set to Specified, type here the number of milliseconds (greater than 0) after which messages on this queue expire.
Base Queue Name	* InputQ	The name of the queue to which messages are sent, on the queue manager specified by the Base queue manager name property.
Base Queue Manager Name		The name of the WebSphere MQ queue manager to which messages are sent.
CCSID		The coded character set identifier for use with the WebSphere MQ queue manager.
Native Encoding	☐ Use native encoding	When enabled, native encoding is used. When disabled, the settings for integer, decimal and floating point are used.
Integer Encoding	Normal	If native encoding is not enabled, select whether integer encoding is normal or reversed.

WebSphere Status

January 2, 2003 2:50:07 PM EST

Figure 11-33 WebSphere MQ queue destination object properties

The configuration properties include:

– **JNDI name:** The JNDI name used to bind the queue into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

– **Persistence:** Specifies the persistence of messages sent to this destination. The possible values are:

- PERSISTENT
- NON PERSISTENT
- APPLICATION DEFINED
- QUEUE DEFINED

The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that instructs the WebSphere MQ JMS provider to store the message in case the provider fails.

For QUEUE DEFINED persistence, messages on the destination queue have their persistence defined by the WebSphere MQ queue definition properties.

Note: For the full WebSphere MQ JMS provider, messaging queues get created in WebSphere MQ under a queue manager. WebSphere MQ allows you to browse messages on a queue, send a test message, clear messages on a queue, etc.

For the embedded WebSphere JMS provider, messaging queues were created from the definition to a JMS server process. There is no provided mechanism for sending test messages or clearing existing queues.

– **Priority:** Specifies the message priority for this destination queue. The possible values are:

  - APPLICATION DEFINED
  - QUEUE DEFINED
  - SPECIFIED

  The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that defines the message priority.

  For QUEUE DEFINED priority, messages on the destination queue have their priority defined by the WebSphere MQ queue definition properties.

– **Specified Priority:** If the priority is set to `SPECIFIED`, this field determines the message priority for this queue. The 10 levels of priority range from 0 (lowest) to 9 (highest). Messages sent to this queue have the priority value specified by this property.

– **Expiry:** Specifies the expiration timeout for this queue. Possible values are:

  - APPLICATION DEFINED
  - UNLIMITED
  - SPECIFIED

  The default is APPLICATION DEFINED. It is the application that puts the message onto the queue that defines whether the message is destroyed after a period of time.

- **Specified Expiry:** If the expiry property is set to `SPECIFIED`, type here the number of milliseconds (greater than 0) after which messages on this queue expire. Zero indicates that messages never time out.
- **Base Queue Name:** The name of the actual WebSphere MQ messaging queue to which messages are sent.

Note: The Base Queue Name is the actual messaging queue created in WebSphere MQ under a queue manager. For the embedded WebSphere JMS provider, messaging queues get automatically created for you when you define them to the JMS server. With full WebSphere MQ you need to create these messaging queues yourself.

WebSphere MQ allows you to manage queues using tools such as the WebSphere MQ Explorer wizard. This is not possible with the WebSphere JMS provider.

- **Base Queue Manager Name:** The name of the WebSphere MQ queue manager hosting the queue specified by the Base Queue Name property.
- **CCSID:** The coded character set identifier used to connect to the WebSphere MQ provider queue manager. This coded character set identifier must be one of the CCSIDs supported by WebSphere MQ.

  Note: For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

- **Native Encoding:** Check this box to indicate that the queue destination is to use native encoding to represent numeric values in the message data. If native encoding is not enabled (the default), you need to specify the properties for Integer, Decimal and Floating Point encoding.

  Note: For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

- **Integer Encoding:** If native encoding is not enabled, select whether the encoding defined for binary integers is Normal or Reversed. The default value is Normal.
- **Decimal Encoding:** If native encoding is not enabled, select whether the encoding defined for decimal integers is Normal or Reversed. The default value is Normal.

- **Floating Point Encoding**. If native encoding is not enabled, select whether the encoding defined for floating-point numbers is IEEENormal, IEEEReversed or S390. The default value is IEEE_Normal.

  Note: For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

- **Target Client:** Whether the receiving application is a JMS-compliant application (JMS) or not (WebSphere MQ).
- **Queue Manager Host:** The host name of the machine where the queue manager for this queue destination resides.
- **Queue Manager Port:** The port number used by the queue manager hosting this queue.
- **Server Connection Channel Name:** The name of the channel used for connection to the WebSphere MQ queue manager hosting this queue.
- **User ID:** The user ID that is used for authentication when connecting to the queue manager hosting the queue destination. If a value is specified for the User Name property, a value must also be specified for the Password property.
- **Password:** The password that, together with the User Name property, is used for authentication when connecting to the queue manager hosting the queue destination.

  Note:

  If you want WebSphere Application Server to try to use the WebSphere MQ queue manager's remote administration utilities to create the queue, configure the WebSphere MQ Queue Connection properties.

  These properties, including Queue Manager Host, Queue Manager Port, Server Connection Channel, User Name and Password are not mandatory.

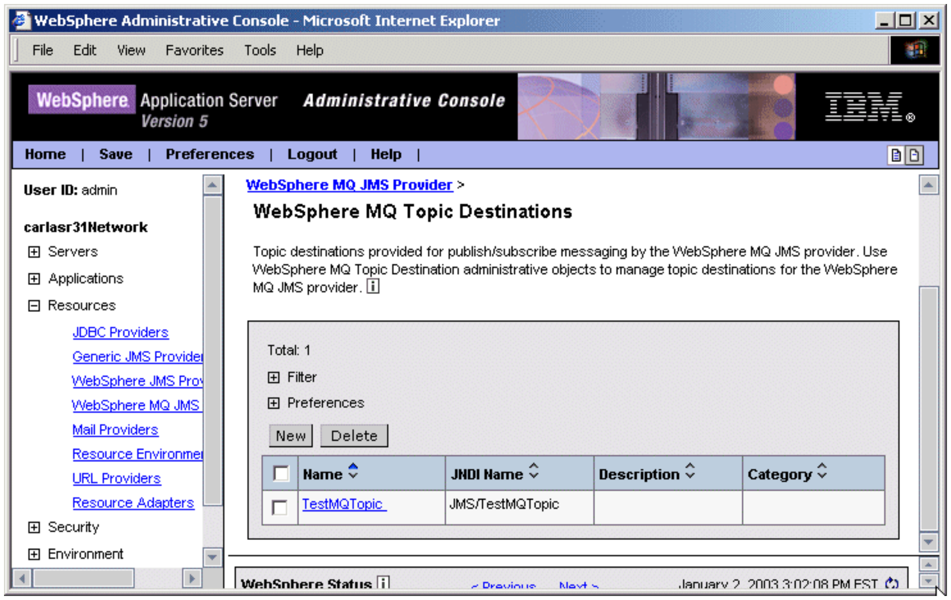
5. Click **OK** when finished and save the configuration.
6. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configuring topic destinations

Complete the following steps to configure the properties of a topic destination administrative object, which represents a topic served by the publish/subscribe JMS provider installed with WebSphere MQ.

1. In the navigation tree, expand **Resources -> WebSphere MQ JMS Providers**.
2. Select **Scope** and click **Apply**. For a discussion on the scope setting, see "Configuring a queue connection factory" on page 510.
3. Click **WebSphere MQ Topic Destinations** in the Additional Properties table.

   This displays WebSphere MQ topic destination objects available to application servers under the selected scope. These objects represent topics served by a publish/subscribe JMS provider installed with WebSphere MQ. This is shown in Figure 11-34.



WebSphere MQ Topic Destinations

Topic destinations provided for publish/subscribe messaging by the WebSphere MQ JMS provider. Use WebSphere MQ Topic Destination administrative objects to manage topic destinations for the WebSphere MQ JMS provider.

Total: 1

	Name	JNDI Name	Description	Category
☐	TestMQTopic	JMS/TestMQTopic		

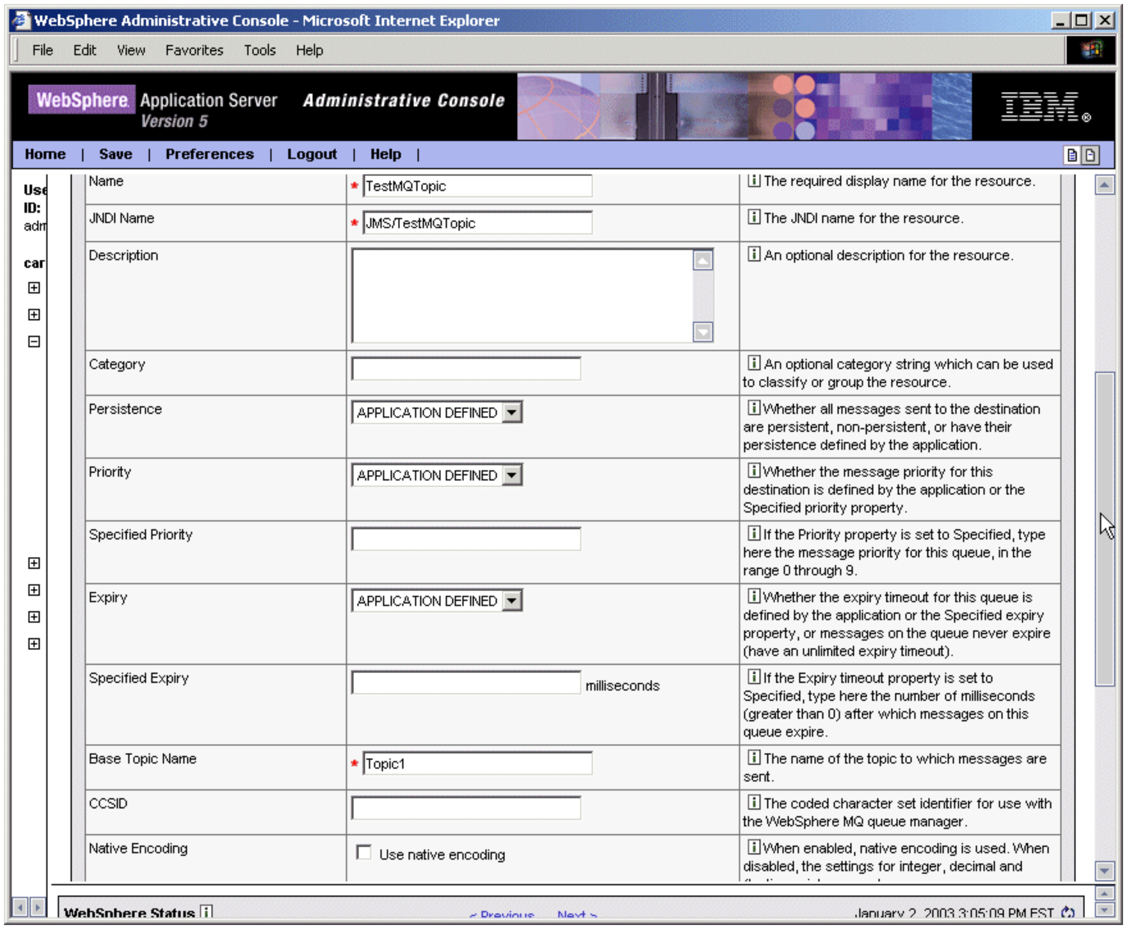
January 2, 2003 3:02:08 PM EST

Figure 11-34 WebSphere MQ topic destination administered objects

In this example we have one WebSphere MQ topic destination object, TestMQTopic, defined. This topic destination object represents a topic served by the publish/subscribe provider installed with WebSphere MQ.

4. To create a new topic destination object, click **New**, or to change the properties of an existing topic destination object, click one of the topic destination objects displayed.

   Figure 11-35 shows the configuration page for topic destination objects.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Use
ID:
adm
car

Name	* TestMQTopic	The required display name for the resource.
JNDI Name	* JMS/TestMQTopic	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Persistence	APPLICATION DEFINED	Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	Whether the message priority for this destination is defined by the application or the Specified priority property.
Specified Priority		If the Priority property is set to Specified, type here the message priority for this queue, in the range 0 through 9.
Expiry	APPLICATION DEFINED	Whether the expiry timeout for this queue is defined by the application or the Specified expiry property, or messages on the queue never expire (have an unlimited expiry timeout).
Specified Expiry	milliseconds	If the Expiry timeout property is set to Specified, type here the number of milliseconds (greater than 0) after which messages on this queue expire.
Base Topic Name	* Topic1	The name of the topic to which messages are sent.
CCSID		The coded character set identifier for use with the WebSphere MQ queue manager.
Native Encoding	☐ Use native encoding	When enabled, native encoding is used. When disabled, the settings for integer, decimal and

Figure 11-35 WebSphere MQ topic destination object properties

The configuration properties available include:

– **JNDI name:** The JNDI name used to bind the topic destination into the application server’s name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

– **Persistence:** Specifies the persistence of messages sent to this topic destination. The possible values are:

  • PERSISTENT

  - NON PERSISTENT
  - APPLICATION DEFINED
  - QUEUE DEFINED

  The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that instructs the publish/subscribe provider installed with WebSphere MQ to store the message in case the provider or the queue manager fails.

  For QUEUE DEFINED persistence, messages on the topic destination queue have their persistence defined by the WebSphere MQ queue definition properties.

- **Priority**

  Specifies the message priority for this topic destination. The possible values are:

  - APPLICATION DEFINED
  - QUEUE DEFINED
  - SPECIFIED

  The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that defines the message priority.

  For QUEUE DEFINED priority, messages on the topic destination queue have their priority defined by the WebSphere MQ queue definition properties.

- **Specified Priority:** If the priority is set to `SPECIFIED`, type here the message priority for this topic. The 10 levels of priority range from 0 (lowest) to 9 (highest). Messages sent to this topic have the priority value specified by this property.
- **Expiry:** Specifies whether the expiration timeout for this topic destination is defined by the application, APPLICATION DEFINED, whether we specify the expiration time, SPECIFIED, or whether the message on the topic destination never expires, UNLIMITED. The default is APPLICATION DEFINED. It is the application that puts the message onto the topic destination that defines whether the message is destroyed after a period of time.
- **Specified Expiry:** If the Expiry property is set to `SPECIFIED`, this field determines the number of milliseconds (greater than 0) after which messages on this topic destination expire. Zero indicates that messages never timeout.
- **Base Topic Name:** The name of the publish/subscribe provider topic to which messages are sent.

– **CCSID:** The coded character set identifier used to connect to the queue manager hosting the publish/subscribe provider. This coded character set identifier must be one of the CCSIDs supported by WebSphere MQ.

> **Note:** For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *MQSeries System Administration* document, SC33-1873-02.

– **Native Encoding:** Select this check box to indicate that the topic destination should use native encoding to represent numeric values in the message data. If native encoding is not enabled (the default), you need to specify the properties for Integer, Decimal and Floating Point encoding.

> **Note:** For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

– **Integer Encoding:** If native encoding is not enabled, select whether the encoding defined for binary integers is Normal or Reversed. The default value is Normal.

– **Decimal Encoding:** If native encoding is not enabled, select whether the encoding defined for decimal integers is Normal or Reversed. The default value is Normal.

– **Floating Point Encoding:** If native encoding is not enabled, select whether the encoding defined for floating-point numbers is IEEENormal, IEEEReversed or S390. The default value is IEEE_Normal.

> **Note:** For information about encoding properties, see the *MQSeries Using Java* document, SC34-5456-08.

– **Target Client:** Whether the receiving application is a JMS-compliant application (JMS) or not (WebSphere MQ).

– **Broker Durable Subscription Queue:** The name of the broker's queue from which durable subscription messages are retrieved.

– **Broker CC Durable Subscription Queue:** The name of the broker's queue from which durable subscription messages are retrieved for a ConnectionConsumer.

> **Note:** The above queues are WebSphere MQ queues managed by the broker queue manager.

5. Click **OK** when finished and save the configuration.
6. To have the changed configuration take effect, stop then restart the application servers using the resource.

11.8.7 Configuring resources for generic JMS provider

If you use a JMS provider other than the embedded WebSphere JMS provider or a WebSphere MQ JMS provider, you can still use the administrative console to register JMS destinations and JMS connection factories into the WebSphere name space.

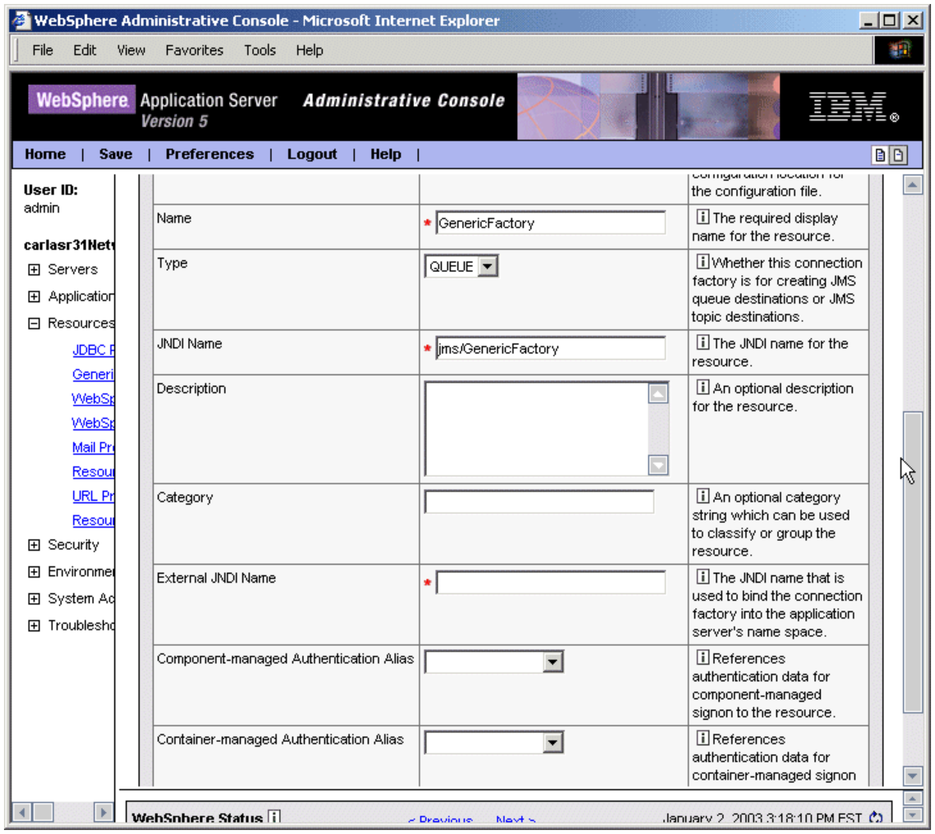
Note: For a detailed description of how WebSphere V5 supports generic JMS providers, see Chapter 11, “Asynchronous messaging” on page 451.

Configuring a JMS connection factory

To configure properties for a JMS connection factory object that will be used to connect to a generic JMS provider, complete these steps:

1. In the navigation tree, expand **Resources -> Generic JMS Providers**.
2. Select **Scope** and click **Apply**.

   For a discussion on the scope setting, see “Configuring a queue connection factory” on page 510.
3. Select the generic JMS provider for which you want to configure the connection factory. This displays a table of properties for the JMS provider and links to the types of JMS resources supported.
4. Click **JMS Connection Factories** in the Additional Properties table to display the generic provider connection factory administered objects used to connect to the generic JMS provider.
5. To create a new connection factory object, click **New**, or to change the properties of an existing connection factory, click one of the connection factories displayed. Figure 11-36 on page 544 shows the configuration page for a connection factory object.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Nett
Servers
Application
Resources
JDBC P
Generi
WebSp
WebSp
Mail Pr
Resou
URL Pr
Resou
Security
Environme
System Ac
Troublesh
WebSphere Status
< Previous Next >
January 2, 2003 3:18:10 PM EST

		the configuration file.
Name	* GenericFactory	The required display name for the resource.
Type	QUEUE	Whether this connection factory is for creating JMS queue destinations or JMS topic destinations.
JNDI Name	* jms/GenericFactory	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
External JNDI Name	*	The JNDI name that is used to bind the connection factory into the application server's name space.
Component-managed Authentication Alias		References authentication data for component-managed signon to the resource.
Container-managed Authentication Alias		References authentication data for container-managed signon

Figure 11-36 Generic connection factory properties

The configuration properties available include:

– **Name:** The name by which this connection factory is known for administrative purposes.

– **Type:** Whether the connection factory is used for point-to-point messaging, QUEUE, or for publish/subscribe messaging, TOPIC.

– **JNDI name:** The JNDI name used to bind the connection factory into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.

– **External JNDI Name:** The connection factory JNDI name, as registered in the generic provider's name space.

Note: The connection factory object that holds connection properties with the generic JMS provider is registered with the generic provider name space.

With the administrative console, what we are creating is a local representation of it, that is a connection factory object registered in the application server's name space and bound to the remote object.

The External Initial Context Factory and External Provider URL were provided as part of the configuration of the provider, see 11.8.3, "Managing a generic JMS provider" on page 508.

For the resource lookup to work, the generic provider's InitialContextFactory and JMS client implementation classes will need to be included in the application server's classpath.

For further details, see Chapter 11, "Asynchronous messaging" on page 451.

– **Component-managed/Container-managed Authentication Alias:** Specifies a user ID and password to be used to authenticate connection to a JMS provider. The entry references authentication data defined in the J2C authentication data entries (in the Additional Properties table).

Other properties configurable in the Additional Properties table are connection pool settings, session pool settings, and J2C authentication data entries.

Note: JMS connection and session resources are managed by the J2C connection manager.

6. Click **OK** when finished and save the configuration.
7. To have the changed configuration take effect, stop then restart the application servers using the resource.

Configuring a JMS destination

Complete the following steps to configure the properties of a JMS destination object that represents a resource hosted by generic JMS provider:

1. In the navigation tree, expand **Resources -> Generic JMS Providers**.
2. Select **Scope** and click **Apply**. For a discussion on the scope setting, see "Configuring a queue connection factory" on page 510.

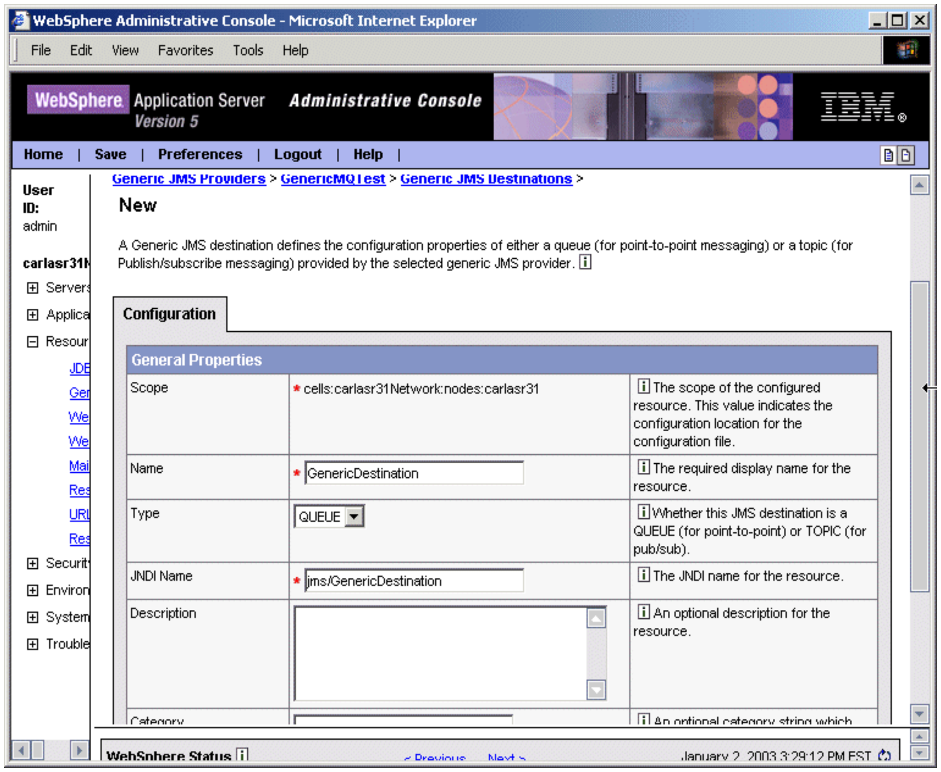
3. Select the generic JMS provider for which you want to configure the destination object.

   This displays a table of properties for the JMS provider and links to the types of JMS resources supported.

4. Click **JMS Destinations** in the Additional Properties table. This destination object represents a destination resource hosted by the generic JMS provider.

5. To create a new destination object, click **New**, or to change the properties of an existing destination object, click one of the destination objects displayed.

   Figure 11-37 shows the properties for the GenericDestination destination object.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
WebSphere Application Server Version 5
Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31N
Servers
Applica
Resour
Securit
Environ
System
Trouble
WebSphere Status
< Previous Next >

Generic JMS Providers > GenericMQTest > Generic JMS Destinations >

New

A Generic JMS destination defines the configuration properties of either a queue (for point-to-point messaging) or a topic (for Publish/subscribe messaging) provided by the selected generic JMS provider.

Configuration

General Properties		
Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* GenericDestination	The required display name for the resource.
Type	QUEUE	Whether this JMS destination is a QUEUE (for point-to-point) or TOPIC (for pub/sub).
JNDI Name	* jms/GenericDestination	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which

January 2, 2003 3:29:12 PM EST

Figure 11-37 Generic destination object properties

The configuration properties include:

– **Name:** The name by which the destination object is known for administrative purposes.

- **Type:** Whether this JMS destination object represents a messaging QUEUE for point-to-point messaging or a TOPIC for publish/subscribe messaging.
- **JNDI name:** The JNDI name used to bind the destination into the application server's name space. As a convention, use the form jms/Name, where Name is the logical name of the resource.
- **External JNDI Name:** The destination JNDI name, as registered in the generic provider's name space.

> **Note:** The destination object (queue or topic) that holds properties for the JMS provider queue or topic messaging resource is registered with the generic provider name space.
>
> With the administrative console, what we are creating is a local representation of it, that is a destination registered in the application server's name space and bound to the remote object.

6. Click **OK** when finished and save the configuration.

To have the changed configuration take effect, stop then restart the application servers using the resource.

11.9 References and resources

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter, "Implementing WebSphere J2EE applications that use JMS" and "Administering WebSphere JMS support".

  `http://www.ibm.com/software/webservers/appserv/infocenter.html`
- Java Message Service API documentation

  `http://java.sun.com/products/jms`
- MA88: MQSeries® classes for Java and MQSeries classes for Java Message Service.

  `http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html`

12

Configuring WebSphere resources

Resource providers are a class of objects that provide resources needed by running Java applications, and J2EE applications in particular. For example, if an application requires database access through a data source, you would need to install a JDBC data source provider and then configure a data source to be used by your application.

This chapter discusses the following application server resource providers:

- JDBC providers for obtaining relational database data sources.
- JavaMail providers for obtaining mail sessions.
- URL providers for obtaining URLs.
- J2C providers for obtaining Enterprise Information Systems access.
- Environment providers for obtaining environment resource factories.

Note: Configuring JMS resources is discussed in Chapter 11, “Asynchronous messaging” on page 451.

12.1 JDBC resources

The JDBC API provides a programming interface for data access of relational databases from the Java programming language. The JDBC 2.0 API is comprised of two packages:

- The java.sql package, which is the JDBC 2.0 core API.
- The javax.sql package, which is the JDBC 2.0 Standard Extension API. This package provides data source and connection pooling functionality.

In the next sections we explain how to create and configure data source objects for use by JDBC applications. This is the recommended way of getting a connection to a database, and the only way if you are looking to use connection pooling and or distributed transactions.

12.1.1 What are JDBC providers and data sources?

A data source represents a real-world data source, such as a relational database. When a data source object has been registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the DataSource object. This makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property in the data source. None of the code using that data source needs to be touched.

Once a data source has been registered with an application server's JNDI name space, application programmers can use it to make a connection to the data source it represents.

The connection will usually be a pooled connection, that is once the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

Data source classes and JDBC drivers are implemented by the data source vendor. By configuring a JDBC provider, we are providing information about the set of classes used to implement the data source and the database driver, that is it provides the environment settings for the DataSource object.

Note: A driver may be written purely in the Java programming language or in a mixture of the Java programming language and the Java Native Interface (JNI) native methods.

In the next sections we describe how to create and configure data source objects, as well as how to configure the connection pools used to serve connections from the data source.

12.1.2 WebSphere support for data sources

The programming model is as follows:

1. An application retrieves a DataSource object from the JNDI naming space.
2. After the DataSource object is obtained, the application code calls getConnection() on the data source to get a Connection object. The connection is a pooled connection, that is it is obtained from a pool of connections.
3. Once the connection is acquired, the application then sends SQL queries or updates to the database.

WebSphere Application Server V5 provides two types of data sources, each differentiated by how the connections are handled:

- WebSphere Version 4 data source
- WebSphere Version 5 data source

Version 4 data source

WebSphere Application Server V4 provided its own JDBC connection manager to handle connection pooling and JDBC access. This support is included with WebSphere Application Server V5 to provide support for J2EE 1.2 applications. If an application chooses to use a Version 4 data source, the application will have the same connection behavior as in Version 4 of the Application Server.

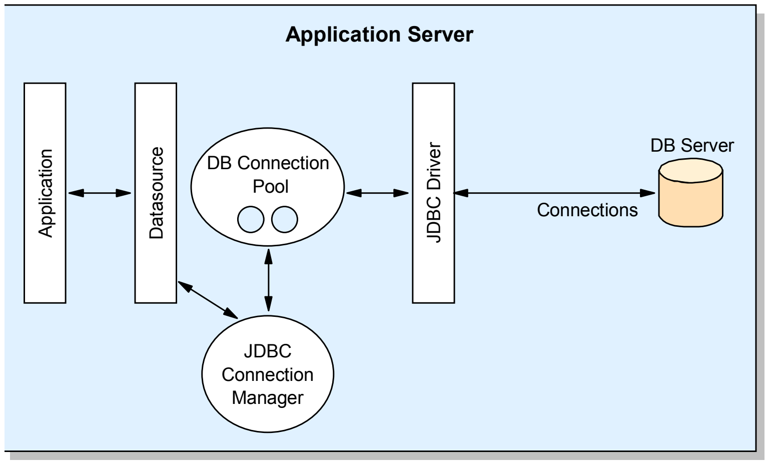


Figure 12-1 Connection pooling in WebSphere Version 4

WebSphere 5 data source

In WebSphere Application Server V5, connection pooling is provided by two parts, a JCA Connection Manager, and a relational resource adapter.

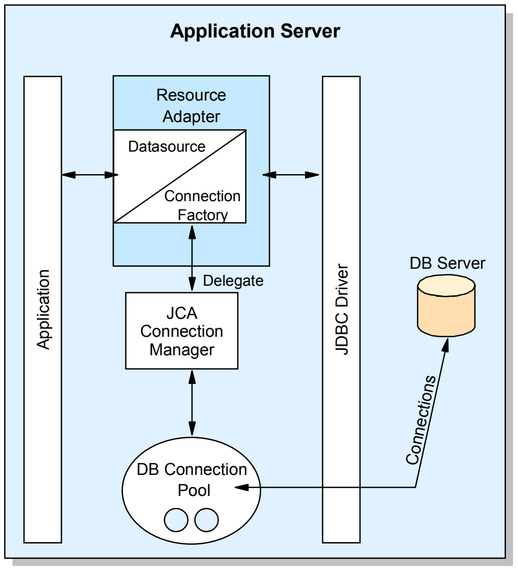


Figure 12-2 Resource adapter in J2EE connector architecture

The JCA Connection Manager provides the connection pooling, local transaction, and security supports. The relational resource adapter provides both JDBC wrappers and JCA CCI implementation that allows BMP, JDBC applications, and CMP beans to access the database. Figure 12-3 shows the relational resource adapter model.

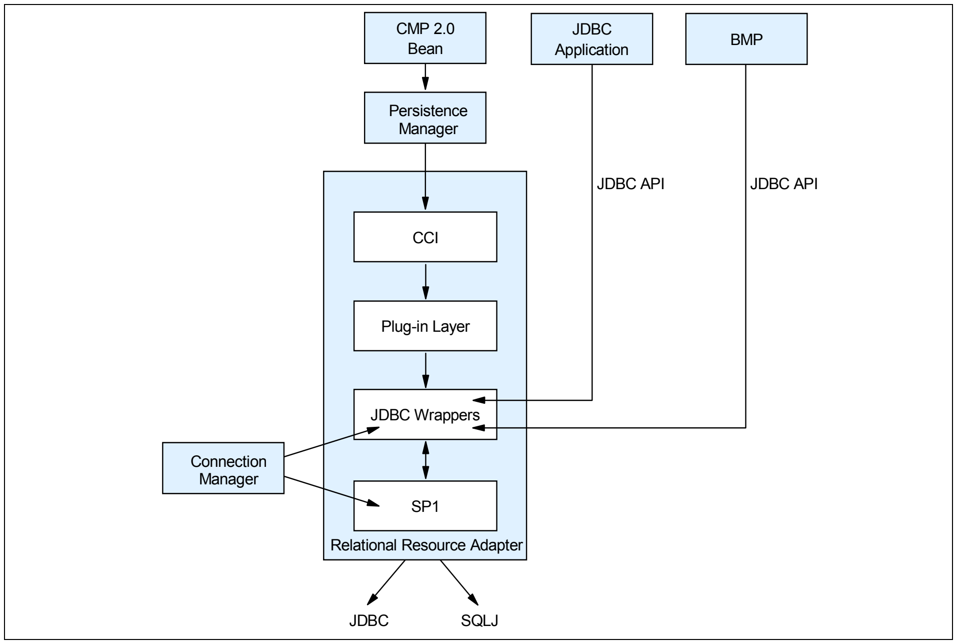


Figure 12-3 Persistence resource adapter model

Note: WebSphere provides a Persistence Resource Adapter to provide relational persistence services to the EJB beans that are deployed in the J2EE 1.3 application as well as providing database access to BMP and JDBC applications. The Persistence Resource Adapter has two components: the persistence manager, which supports the new EJB 2.0 CMP persistence model, and the relational resource adapter.

The persistence resource adapter code is included in the following Java packages:

- com.ibm.ws.rsadapter.cci - contains CCI implementation and JDBC wrappers.
- com.ibm.ws.rsadapter.spi - contains SPI implementation.
- com.ibm.ws.rsadapter.jdbc - contains all the JDBC wrappers.
- com.ibm.websphere.rsadapter - DataStoreHelper, WSCallerHelper and DataAccessFunctionSet.

The relational resource adapter is the EJB 2.0 Persistence Manager's persistent vehicle to handle the data access to/from the back-end store. It provides relational persistence services to the EJB beans that are deployed in the J2EE 1.3 applications. The relational resource adapter implementation is based on the J2EE Connector (JCA) specification and implements the JCA CCI and SPI interfaces.

The relational resource adapter that is available with Network Deployment only provides JDBC data access to a relational database, but potentially the relational resource adapter can be used for other forms of data access.

The following database platforms are supported for JDBC: DB2 family, Oracle, Sybase, Informix®, SQLServer, Cloudscape and third-party vendor JDBC data source using SQL99 standards.

If an application chooses to use a Version 5 data source, the data source will use the JCA connector architecture to get to the relational database. Although there is no difference between the existing WebSphere JDBC support versus the new support in terms of application development, there will be some connection behavior changes because of different architectures.

For an EJB the sequence is as follows:

1. An EJB performs a JNDI lookup of a data source connection factory and issues a getConnection() request.
2. The connection factory delegates the request to a connection manager.
3. The connection manager looks for an instance of a connection pool in the application server. If no connection pool is available, then the manager uses the ManagedConnectionFactory to create a physical (nonpooled) connection.

So from a JDBC application point of view, there is no difference between using a Version 4 data source or a Version 5 data source. It is the implementation of the data source that changes.

For more details, refer to the J2EE Connector Architecture specification at:

`http://java.sun.com/j2ee/connector/`

12.1.3 Data source usage guidelines

The data source that can be used is determined by whether the application is a J2EE 1.2 or a J2EE 1.3 application, and whether it uses EJB 1.1 modules or EJB 2.0 modules. If you try to use a Version 5 data source with an EJB 1.1 module, the application will fail to start or will start with errors.

The guidelines for data source use are:

- **J2EE 1.2 applications**

  All EJB beans, JDBC applications, or Version 2.2 servlets must use the Version 4 data source.

- **J2EE 1.3 applications**

  a. EJB 1.x modules (1.1 deployment descriptor) must use the Version 4 data source.

  b. EJB 2.0 modules (2.0 deployment descriptor), including CMP Version 2.0 and 1.x must use the Version 5 data source.

  c. JDBC applications and Version 2.3 servlets must use the Version 5 data source.

Table 12-1 outlines the possible combinations of EJB modules and data sources supported by WebSphere Application Server V5.

Table 12-1 Version 4 versus Version 5 data sources

	EJB 1.1	**EJB 2.0**
J2EE 1.2 deployment descriptor	Version 4 data sources	N/A
J2EE 1.3 deployment descriptor	Version 5 data sources (recommended) or Version 4 data sources	Version 5 data sources

12.2 Creating a data source

The following steps are involved in creating a data source:

1. Create a JDBC provider resource:

   The JDBC provider gives the classpath of the data source implementation class and the supporting classes for database connectivity. This is vendor specific.

2. Create a data source resource:

   The JDBC data source encapsulates the database specific connection settings.

Note: The JDBC provider needs to be created before we can create the data source.

12.2.1 Creating a JDBC provider

To create a JDBC provider, complete the following steps from the administrative console:

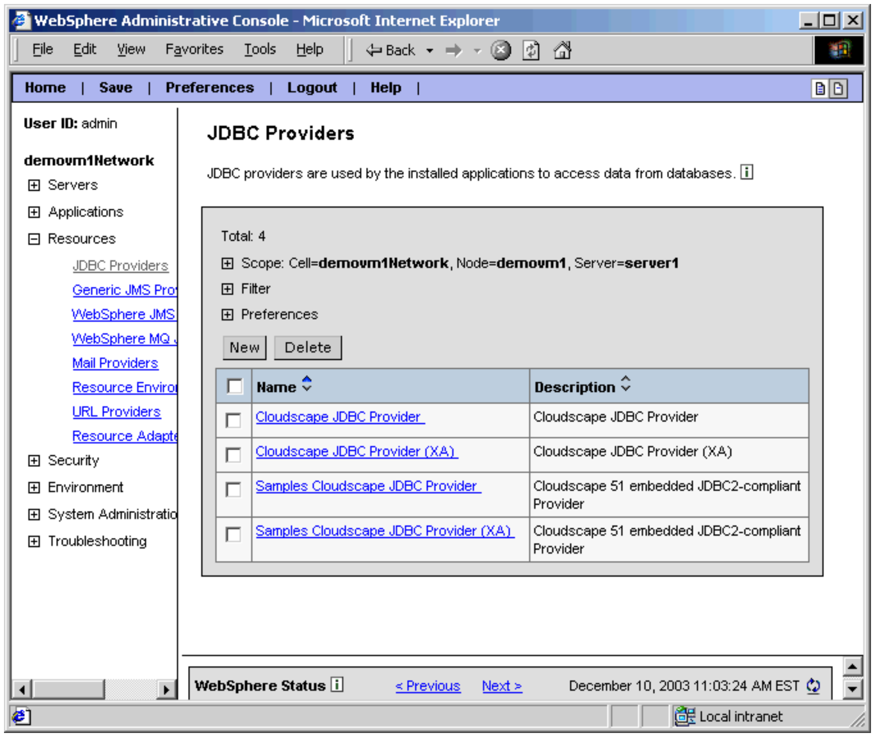
1. Expand **Resources** from the navigation tree.
2. Click **JDBC Providers**.
3. Select **Scope** and click **Apply**.

Note: JDBC resources, for example data sources, are created at a specific scope level. The data source scope level is inherited from the JDBC provider. For example, if we create a JDBC provider at the node level, and then create a data source using that JDBC provider, the data source will inherit:

- The JDBC provider settings, such as classpath, implementation class, etc.
- The JDBC provider scope level. In this example, if the scope was set to node level, all application servers running on that node will register the data source in their name space.

The resources.xml file will also get updated at the node and application server level.

The administrative console will now show all the JDBC providers created at that scope level. In Figure 12-4 on page 557, you can see that there are three JDBC providers defined at the node level.



JDBC Providers

JDBC providers are used by the installed applications to access data from databases.

Total: 4

Scope: Cell=**demovm1Network**, Node=**demovm1**, Server=**server1**

Filter

Preferences

	Name	Description
☐	Cloudscape JDBC Provider	Cloudscape JDBC Provider
☐	Cloudscape JDBC Provider (XA)	Cloudscape JDBC Provider (XA)
☐	Samples Cloudscape JDBC Provider	Cloudscape 51 embedded JDBC2-compliant Provider
☐	Samples Cloudscape JDBC Provider (XA)	Cloudscape 51 embedded JDBC2-compliant Provider

December 10, 2003 11:03:24 AM EST

Figure 12-4 JDBC providers

4. Select **New** to create a new JDBC provider.
5. Use the drop-down list to select the type of provider you want to create.

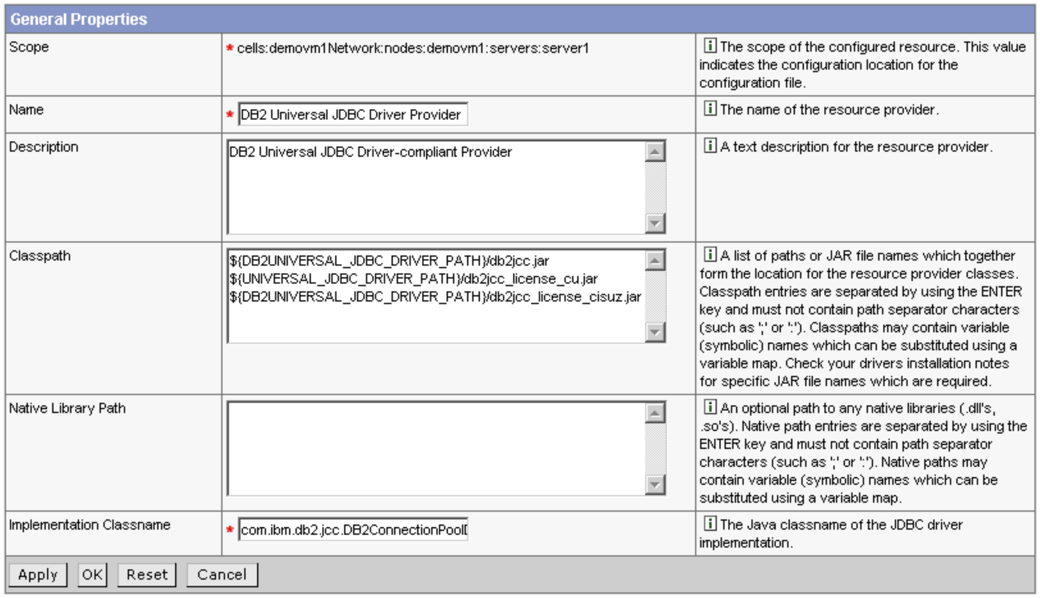
   For example, use the DB2 Universal JDBC Driver Provider option if you want your data source to connect to a DB2 database using connection pooling.

   Connection pooling is a mechanism whereby when an application closes a connection, that connection is recycled rather than being destroyed. Because establishing a connection is an expensive operation, reusing connections can improve performance dramatically by cutting down on the number of new connections that need to be created.

   Use the DB2 Universal JDBC Driver Provider (XA)option if you want your data source to connect to a DB2 database using connection pooling, and the connections need to be used in distributed transactions (two-phase commit transactions). A connection capable of being used in a distributed transaction can also be used for a non- distributed transaction.

   If the list of supported JDBC provider types does not include the JDBC provider that you wish to use, select the user-defined JDBC provider. You will need to consult the vendor's documentation for information on specific properties that might be required.

6. Click **Apply**. The settings page for your JDBC provider appears. Figure 12-5 shows the configuration page for a DB2 JDBC Provider.



General Properties

Scope	* cells:demovm1Network:nodes:demovm1:servers:server1	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* DB2 Universal JDBC Driver Provider	The name of the resource provider.
Description	DB2 Universal JDBC Driver-compliant Provider	A text description for the resource provider.
Classpath	${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar	A list of paths or JAR file names which together form the location for the resource provider classes. Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.
Native Library Path		An optional path to any native libraries (.dll's, .so's). Native path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.
Implementation Classname	* com.ibm.db2.jcc.DB2ConnectionPoolI	The Java classname of the JDBC driver implementation.

Figure 12-5 JDBC provider properties

7. Enter the JDBC provider properties.

   As shown in Figure 12-5, the JDBC provider general properties are:

   – **Name:** A name for the provider. It is recommended you enter a name that is suggestive of the database product you are using.

   – **Description:** A description of the provider, for your administrative records.

   – **Classpath:** A list of paths or JAR file names which together form the location for the resource provider classes. For example c:\ibm\sqllib\java\db2java.zip if the data source connects to DB2.

Note:

- The use of WebSphere environment variables gives you more flexibility. For example, if you selected *Cell* as the scope level, and you specified the *Classpath* as per above, this provider would only work on those nodes in which the db2java.zip file is installed to the same directory structure. Applications running on UNIX nodes for example, wouldn't be able to use this JDBC provider's data source.

  Refer to 8.6.1, "Using variables" on page 277 for more information on WebSphere environment variables.
- Classpath entries need to be separated by the <Enter> key.

  – **Native Library Path:** An optional path to any native libraries. Entries are required if the JBDC provider chosen uses non-Java (native) libraries.

  – **Implementation Classname:** The name of the Java data source class used to connect to the database, as provided by the database vendor. This is provided for you when you select the type of JDBC provider.

    For example, the one-phase commit protocol classes for DB2 and Oracle are:

    • DB2: COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource

    • Oracle: oracle.jdbc.pool.OracleConnectionPoolDataSource

    This class must be available in the driver file specified by **Classpath** property above.

8. Click **OK** when finished and save the configuration.

Tip: To make a data source available on multiple nodes using different directory structures, complete the following steps using the administrative console:

1. Define the JDBC provider at the cell scope. Use WebSphere environment variables for the classpath and native path.
2. Create the data source that uses this JDBC provider at the cell scope. All files defined at the cell scope are replicated to every node in the cell.
3. For the paths to the driver on each node to be unique, use a variable to specify the driver location and have that variable be defined differently on each node.

   For example, ${DRIVER_PATH} can be used for the classpath in the provider definition. You can then define a variable called ${DRIVER_PATH} at the cell scope to act as a default driver location. Then you can override that variable on any node by defining ${DRIVER_PATH} at the node scope. The node definition takes precedence over the cell level definition.

12.2.2 Creating the JDBC data source

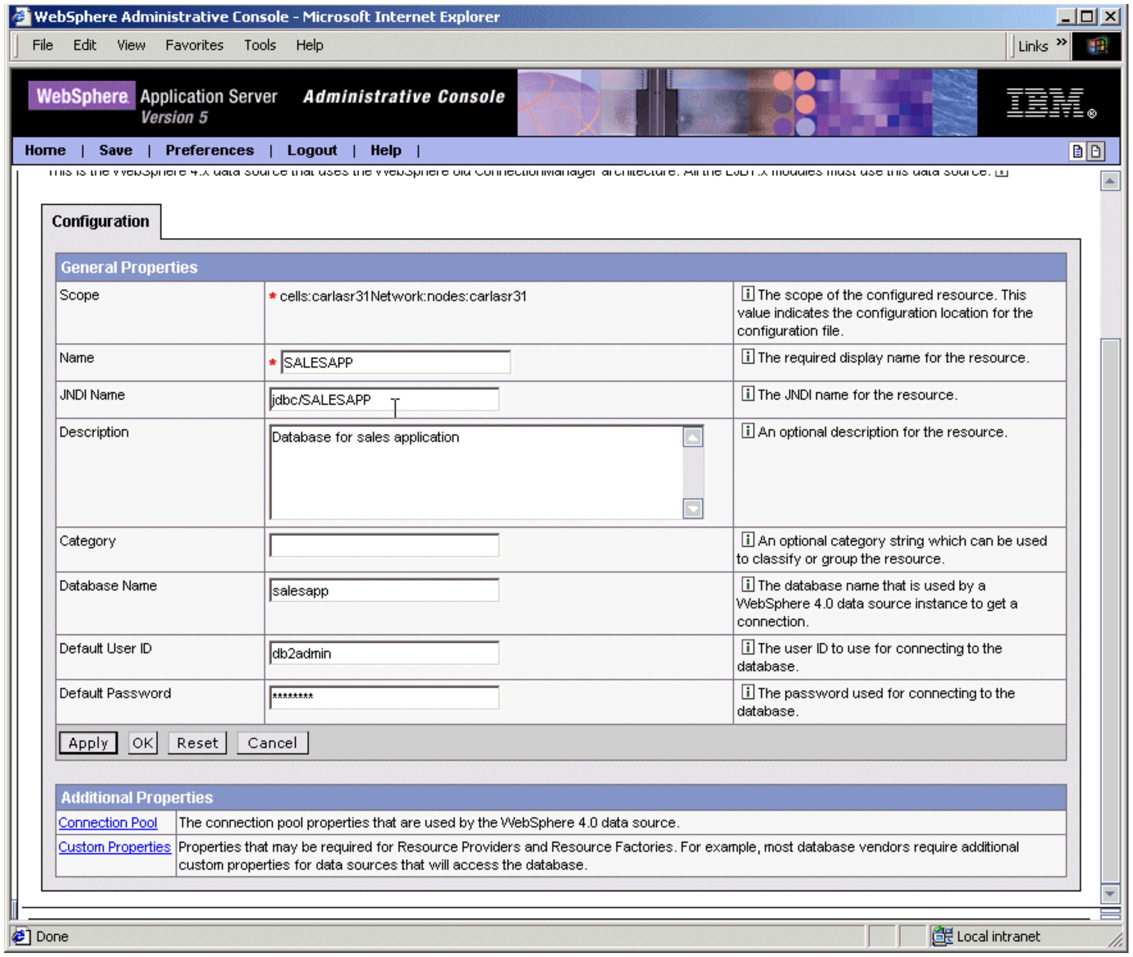
After creating and configuring the JDBC provider, you can now create a data source under it. The data source will use the JDBC provider classes to connect to the database.

Creating a Version 4 data source

To create a Version 4 data source, complete these steps:

1. Expand **Resources** from the navigation tree.
2. Click **JDBC Providers**.
3. Select **Scope** and click **Apply**.
4. Select the JDBC provider to be used by your Version 4 data source.

   This brings up the configuration window for that JDBC provider.
5. Select **Data Sources (Version 4)** in the Additional Properties table. Any Version 4 data sources defined for the selected scope will be listed.
6. Click **New** to create a new data source, or select an existing one to modify the data source properties. The Version 4 data source properties are shown in Figure 12-6 on page 561:



WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

This is the WebSphere 4.x data source that uses the WebSphere old ConnectionManager architecture. All the EJB 1.x modules must use this data source.

Configuration

General Properties

Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* SALESAPP	The required display name for the resource.
JNDI Name	jdbc/SALESAPP	The JNDI name for the resource.
Description	Database for sales application	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Database Name	salesapp	The database name that is used by a WebSphere 4.0 data source instance to get a connection.
Default User ID	db2admin	The user ID to use for connecting to the database.
Default Password	********	The password used for connecting to the database.

Apply OK Reset Cancel

Additional Properties

Connection Pool	The connection pool properties that are used by the WebSphere 4.0 data source.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Done Local intranet

Figure 12-6 Version 4 data source properties

- **Name:** A name by which to administer the data source. Use a name that is suggestive of the database you will use to store data, such as SampleDataSource, where SAMPLE is the database name.
- **JNDI Name:** The data source's name as registered in the application server's name space. When installing an application that contains modules with JDBC resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the Name property prefixed with "jdbc/".

- **Description:** A description of the data source, for your administrative records.
- **Category:** Specifies a category that you can use to classify or group the data source.
- **Database Name:** The name of the database that this data source will access.
- **Default User ID:** The user name for connecting to the database when no user ID and password pair is specified by the application. If the user ID is specified, the password must also be specified.
- **Default Password:** The password for connecting to the database when no user ID and password pair is specified by the application. If the password is specified, the user ID must also be specified.

7. In the Additional Properties table (see Figure 12-6 on page 561), select **Custom Properties**. Some database vendors might require additional custom properties for data sources that will access their database. These are a set of name-value pairs describing properties of the data source. Check the database vendor documentation.

   For example you would need to configure the “URL” property for the Oracle thin driver provider in order to provide database connection details:

   - **Name**: URL
   - **Value**: jdbc:oracle:thin:@<hostname>:<port number>:<databasename>

8. Click **OK** when finished and save the configuration.

Configuring connection pooling for a Version 4 data source

Establishing JDBC connections is resource expensive. Performance can be improved significantly when connection pooling is used. Connection pooling means that connections are reused rather than created each time a connection is requested. For a Version 4 data source the connection pooling is handled by the WebSphere JDBC connection manager.

Connection pooling is transparent to the application. All the application does is to perform a lookup on a JNDI name of a previously registered data source and then issue a getConnection(). If the data source class implements connection pooling, then the client will obtain a connection from a pool of connections.

Note: The use of connection pooling is determined by the JDBC provider class. The connection pool configuration allows you to adjust the connection pool parameters.

To configure the connection pool for a Version 4 data source, complete these steps:

1. Expand **Resources** from the navigation tree.
2. Click **JDBC Providers**.
3. Select **Scope** and click **Apply**.
4. Select the JDBC provider used by your Version 4 data source.
5. Select **Data Sources (version 4)** in the Additional Properties table.
6. Select the data source for which you want to configure connection pooling.
7. in the Additional Properties table, select **Connection Pool**. The properties to configure for a Version 4 data source connection pool are:
   - **Minimum Pool Size:** The minimum number of connections to maintain in the pool.

     The minimum pool size can affect the performance of an application. Smaller pools require less overhead when the demand is low because fewer connections are being held open to the database. On the other hand, when the demand is high, the first applications will experience a slow response because new connections will have to be created if all others in the pool are in use.

     Note: The pool does not start out with the minimum number of connections. As applications request connections, the pool grows up to the minimum number. After the pool has reached the minimum number of connections, it does not shrink beyond this minimum unless an exception occurs that requires the entire pool to be destroyed.

   - **Maximum Pool Size:** The maximum number of connections to maintain in the pool. If the maximum number of connections is reached and all connections are in use, additional requests for a connection will wait up to the number of seconds specified in the **Connection timeout** property.

     The maximum pool size can affect the performance of an application. Larger pools require more overhead when demand is high because there are more connections open to the database at peak demand. These connections persist until they are idled out of the pool as specified by the **Idle Timeout** property.

     On the other hand, if the maximum is smaller, there might be longer wait times or possible connection timeout errors during peak times. The database must be able to support the maximum number of connections in the application server in addition to any load that it may have outside of the application server.

- **Connection Timeout:** The maximum number of seconds that an application waits for a connection from the pool before timing out and throwing a ConnectionWaitTimeoutException to the application. This can occur when the pool is at its maximum and all of the connections are in use by other applications for the duration of the wait.

  In addition, there are no connections currently in use that the application can share, because either the user name and password are different or it is in a different transaction. Setting this value to 0 disables the connection timeout.

- **Idle timeout:** The maximum number of seconds that an idle (unallocated) connection can remain in the pool before the connection is removed from the pool and closed.

  Connections need to be idled out of the pool because keeping connections open to the database can cause memory problems with the database in some cases. However, not all connections are idled out of the pool, even if they are older than the idle timeout number of seconds. A connection is not idled if removing the connection would cause the pool to shrink below its minimum size. Setting this value to 0 disables the idle timeout.

- **Orphan Timeout:** The maximum number of seconds that an application can hold a connection without using it before the connection is returned to the pool.

  If there has been no activity on an allocated connection for longer than the orphan timeout number of seconds, the connection is marked for orphaning. After another orphan timeout number of seconds, if the connection still has had no activity, the connection is returned to the pool. If the application tries to use the connection again, it is thrown a StaleConnectionException. Connections that are enlisted in a transaction are not orphaned. Setting this value to 0 disables the orphan timeout.

  The orphan timeout is a provision to handle an application's failure to release connections or other similar failures.

  Note: The actual amount of time before a connection is closed is approximately twice the orphan timeout value.

- **Statement Cache Size:** The maximum number of cached prepared statements to keep per connection.

  Statement caching improves performance. Retrieval of a matching statement from the cache takes less overhead than creating a new prepared statement. The statement cache size does not change the programming model, only the performance of the application. The statement cache size is the number of cached statements for each connection.

– **Auto Connection Cleanup:** Tells the connection pooling software whether or not to automatically close connections from this data source at the end of a transaction.

  The default is false, which indicates that when a transaction is completed, WebSphere connection pooling closes the connection and returns it to the pool. This means that any use of the connection after the transaction has ended results in a StaleConnectionException, because the connection has been closed and returned to the pool.

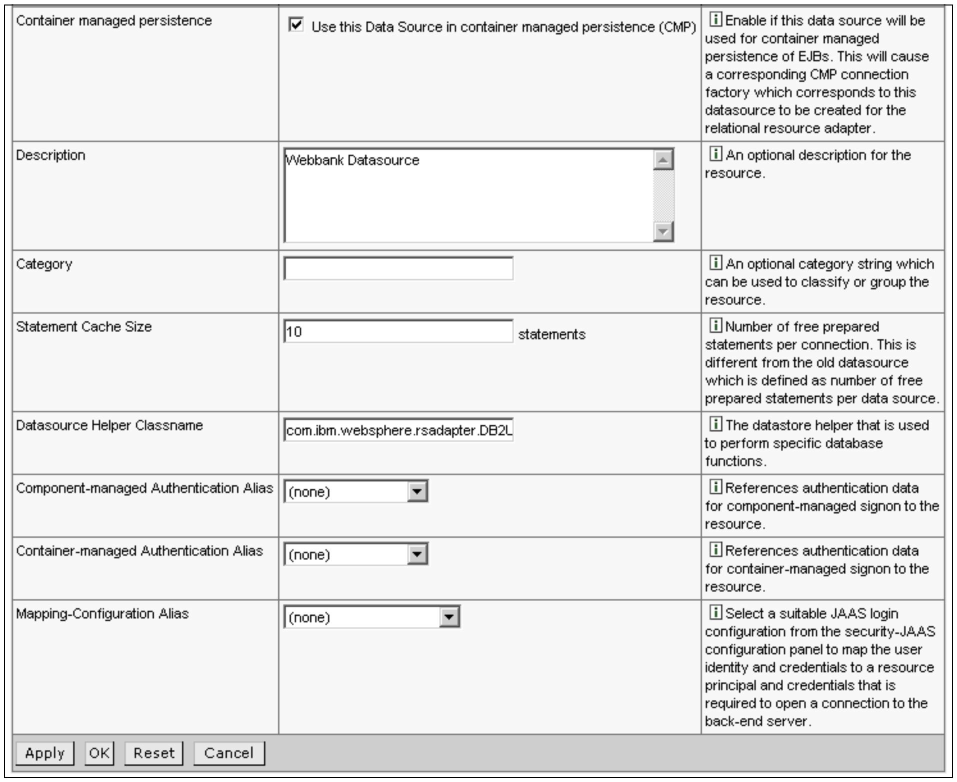
  This mechanism ensures that connections are not held indefinitely by the application. If the value is set to `true`, the connection is not returned to the pool at the end of a transaction. In this case, the application must return the connection to the pool by explicitly calling close(). If the application does not close the connection, the pool can run out of connections for other applications to use. This is different from orphaning connections, because connections in a transaction cannot be orphaned.

8. Click **OK** when finished and save the configuration.

Creating a Version 5 data source

To create a Version 5 data source complete the following steps:

1. Expand **Resources** from the navigation tree.
2. Click **JDBC Providers**.
3. Select **Scope** and click **Apply**.
4. Select the JDBC provider to be used by your data source.
5. Select **Data Sources** in the Additional Properties table of the configuration window (see Figure 12-5 on page 558).
6. Click **New** to create a new data source, or select an existing one to modify the data source properties. The Version 5 data source properties page is shown in Figure 12-7 on page 566:



Container managed persistence	☑ Use this Data Source in container managed persistence (CMP)	Enable if this data source will be used for container managed persistence of EJBs. This will cause a corresponding CMP connection factory which corresponds to this datasource to be created for the relational resource adapter.
Description	Webbank Datasource	An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Statement Cache Size	10 statements	Number of free prepared statements per connection. This is different from the old datasource which is defined as number of free prepared statements per data source.
Datasource Helper Classname	com.ibm.websphere.rsadapter.DB2U	The datastore helper that is used to perform specific database functions.
Component-managed Authentication Alias	(none)	References authentication data for component-managed signon to the resource.
Container-managed Authentication Alias	(none)	References authentication data for container-managed signon to the resource.
Mapping-Configuration Alias	(none)	Select a suitable JAAS login configuration from the security-JAAS configuration panel to map the user identity and credentials to a resource principal and credentials that is required to open a connection to the back-end server.

Figure 12-7 Version 5 data source properties

- **Name:** A name by which to administer the data source. Use a name that is suggestive of the database you will use to store data, such as webbankds.
- **JNDI name:** The data source's name as registered in the application server's name space. When installing an application that contains modules with JDBC resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the Name property prefixed with "jdbc/"

– **Container-managed persistence (CMP):** Specifies if the data source is used for container-managed persistence of EJB beans.

Note: You need to select this check box if your application uses CMP 2.0 beans. It is not necessary to check this box if you only want BMP or JDBC access.

This setting is enabled by clicking the check box. This causes a CMP connection factory corresponding to this data source to be created for the relational resource adapter. The connector factory created has the name datasourcename_CF and is registered in JNDI under the entry eis/datasourcename_CMP.

Tip: You can see the properties of the just created connection factory by selecting **Resources** -> **Manage Resource Adapters -> WebSphere Relational Resource Adapter -> CMP Connection Factories** from the administrative console.

CMP 2.0 EJBs beans will use this CMP connection factory in order to get J2C connections. The Persistence Manager will use this JCA connection factory at runtime. The look up of the connection factory is transparent to the programmer.

In Example 12-1, we see how a CMP 2.0 EJB, webbank/ejb/CustomerAccount, needs its CMPConnectionFactoryBinding bound to the JNDI name of the connection factory, eis/webbankds_CMP, in the example.

Example 12-1 ibm-ejb-jar-bnd.xmi file for webbankEntityEJBs.jar

```
<ejbBindings
xmi:id="CustomerAccount_Bnd"jndiName="webbank/ejb/CustomerAccount">
    <enterpriseBean xmi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#CustomerAccount"/>
    <cmpConnectionFactory xmi:id="CMPConnectionFactoryBinding_2"
jndiName="eis/webbankds_CMP" resAuth="Per_Connection_Factory"/>
  </ejbBindings>
```

Note: Make sure that when you specify the data source to use for a CMP 2.0 EJB, that you ultimately bind your component to the connection factory. This binding is done during application installation. See Chapter 14, “Deploying applications” on page 647 for more information.

Notice also that the resources.xml file, at the selected scope, contains two entries, one for the CMPConnectorFactory object (see Example 12-2) and another one for the DataSource object (see Example 12-3). The same can be seen if you use **dumpNameSpace** to dump the JNDI name space of an application server under the selected scope.

Example 12-2 CMPConnectionFactory resource reference in the resources.xml file

```
<factories xmi:type="resources.jdbc:CMPConnectorFactory"
xmi:id="CMPConnectorFactory_1" name="webbankds_CF"
authMechanismPreference="BASIC_PASSWORD" cmpDatasource="DataSource_1">
      <authDataAlias xsi:nil="true"/>
      <propertySet xmi:id="J2EEResourcePropertySet_1">
        <resourceProperties xmi:id="J2EEResourceProperty_1"
name="TransactionResourceRegistration" type="java.lang.String" value="dynamic"
description="Type of transaction resource registration (enlistment).  Valid
values are either &#34;static&#34; (immediate) or &#34;dynamic&#34;
(deferred)."/>
        <resourceProperties xmi:id="J2EEResourceProperty_2"
name="InactiveConnectionSupport" type="java.lang.Boolean" value="true"
description="Specify whether connection handles support implicit reactivation.
(Smart Handle support). Value may be &#34;true&#34; or &#34;false&#34;."/>
      </propertySet>
</factories>
```

Example 12-3 Data source resource reference in the resources.xml file

```
<factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1"
name="webbankds" jndiName="webbankds" description="New JDBC Datasource"
statementCacheSize="10"
datasourceHelperClassname="com.ibm.websphere.rsadapter.DB2DataStoreHelper"
relationalResourceAdapter="builtin_rra">
      <propertySet xmi:id="J2EEResourcePropertySet_2">
       <resourceProperties xmi:id="J2EEResourceProperty_3" name="databaseName"
type="java.lang.String" value="WEBBANK" description="This is a required
property. The database name. For example, enter sample to make your Data Source
point to jdbc:db2:sample." required="false"/>
        <resourceProperties xmi:id="J2EEResourceProperty_8" name="user"
type="java.lang.String" value="wasadmin" required="false"/>
        <resourceProperties xmi:id="J2EEResourceProperty_9" name="password"
type="java.lang.String" value="wasadmin" required="false"/>
      </propertySet>
      <connectionPool xmi:id="ConnectionPool_4" connectionTimeout="1800"
maxConnections="10" minConnections="1" reapTime="180" unusedTimeout="1800"
purgePolicy="EntirePool"/>
</factories>
```

– **Description:** A description of the data source, for your administrative records.

- **Category:** Specifies a category that you can use to classify or group the data source.
- **Statement Cache Size:** Specifies the number of prepared statements that are cached per connection.
- **Datasource Helper Classname:** Specifies the data store helper that is used to perform database specific functions.

  This is used by the relational resource adapter at runtime. The default DataStoreHelper implementation class is set based on the JDBC driver implementation class, using the structure:

  ```
  com.ibm.websphere.rsadapter.<database>DataStoreHelper
  ```

  For example, if the JDBC provider is DB2, then the default DataStoreHelper class is:

  com.ibm.websphere.rsadapter.DB2DataStoreHelper

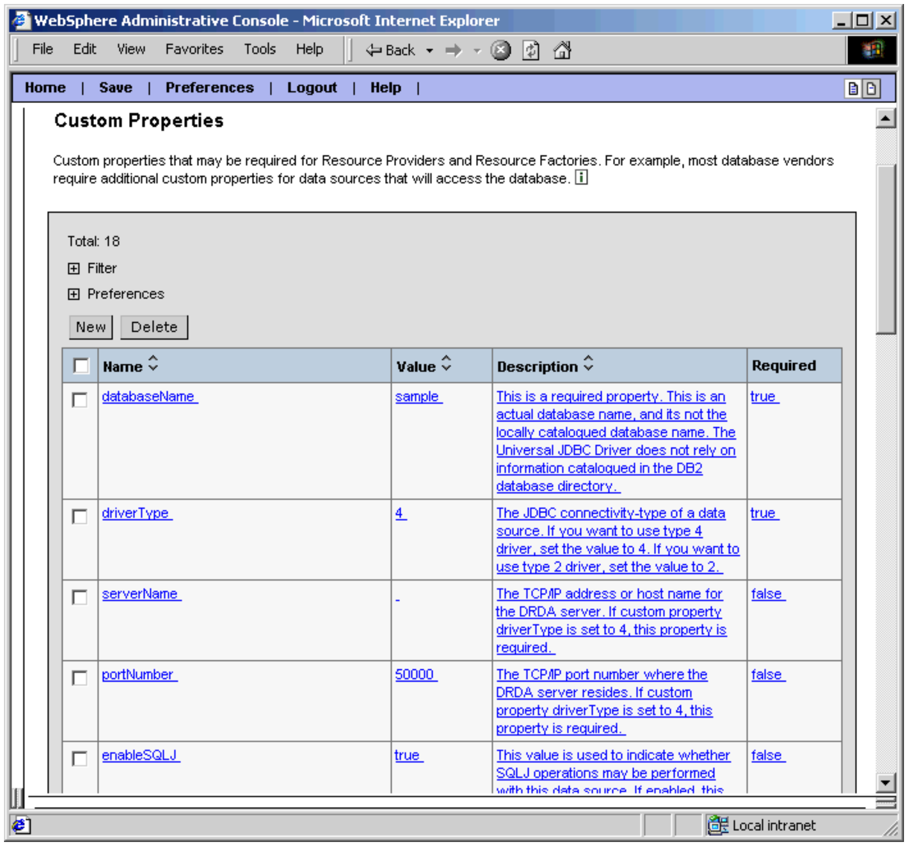
  You can change to value to refer to a specific a subclass of this DataStoreHelper if necessary.
- **Component-managed/Container-managed Authentication Alias:** Specifies a user ID and password to be used by Java 2 Connector security. The entry references authentication data defined in the J2C authentication data entries (in the Additional Properties table).
- **Mapping-Configuration Alias:** Allows users to select from the **Security -> JAAS Configuration -> Application Logins Configuration** list. The DefaultPrincipalMapping JAAS configuration maps the authentication alias to the user ID and password. You may define and use other mapping configurations.

7. Click **OK**.
8. Click **Custom Properties**, in the Additional Properties table, to provide other data source properties, such as:
   - **databaseName:** The name of the database that this data source will access.
   - **user:** The user name for connecting to the database when no user ID and password pair is specified by the application. If the user ID is specified, the password must also be specified.
   - **password:** The password for connecting to the database when no user ID and password pair is specified by the application. If the password is specified, the user ID must also be specified.
   - **preTestSQLString:** Specify the SQL statement to use for connection pre-testing, which helps to ensure that applications obtain valid data sources from connection pools. When connection pre-testing is enabled,

the SQL statement is executed to determine whether the connection is good.

Click **New** to create each one and click **OK** afterwards.

9. Figure 12-8 shows custom properties configured for a Version 5 data source connecting to a DB2 database (databaseName and driverType are required parameters). Other database vendors might require different custom properties. Check the database vendor documentation.



WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Home | Save | Preferences | Logout | Help |
Local intranet

Custom Properties

Custom properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Total: 18

Filter

Preferences

New Delete

	Name	Value	Description	Required
☐	databaseName	sample	This is a required property. This is an actual database name, and its not the locally catalogued database name. The Universal JDBC Driver does not rely on information catalogued in the DB2 database directory.	true
☐	driverType	4	The JDBC connectivity-type of a data source. If you want to use type 4 driver, set the value to 4. If you want to use type 2 driver, set the value to 2.	true
☐	serverName	.	The TCP/IP address or host name for the DRDA server. If custom property driverType is set to 4, this property is required.	false
☐	portNumber	50000	The TCP/IP port number where the DRDA server resides. If custom property driverType is set to 4, this property is required.	false
☐	enableSQLJ	true	This value is used to indicate whether SQLJ operations may be performed with this data source. If enabled, this	false

Figure 12-8 Data Source custom properties

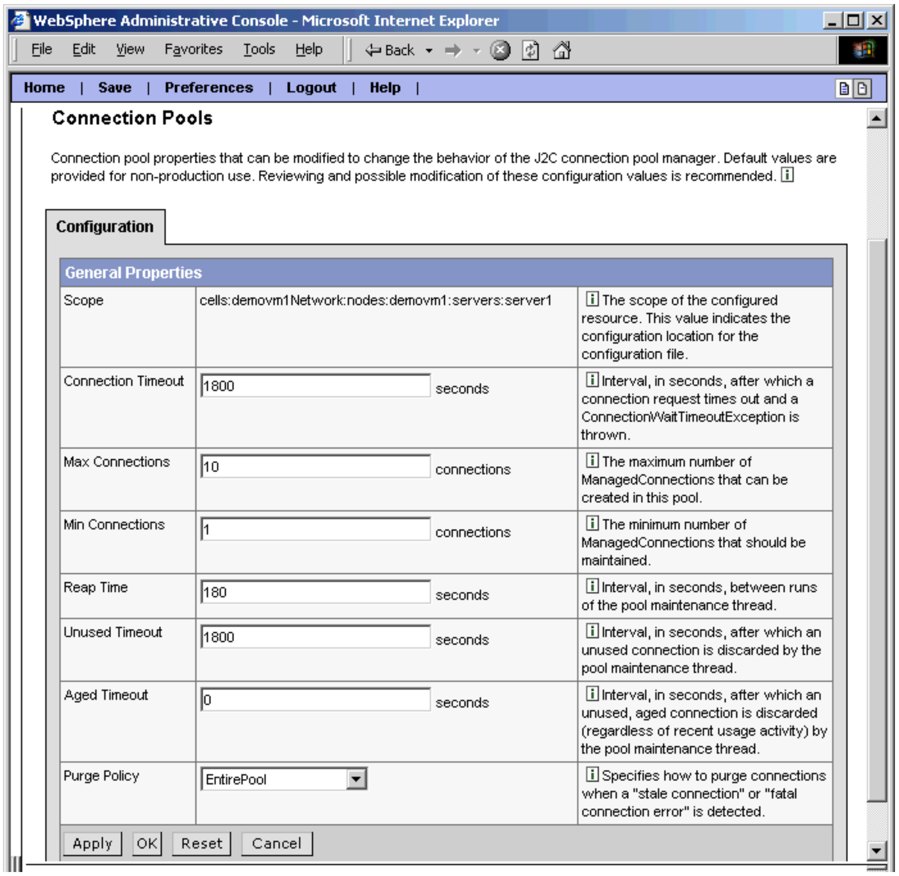
10. Click **OK** when finished and save the configuration.

Configuring connection pooling for a Version 5 data source

To configure the connection pool for a Version 5 data source, complete the following steps:

1. Expand **Resources** from the navigation tree.

2. Click **JDBC Providers**.
3. Select **Scope** and click **Apply**.
4. Select the JDBC provider used by your Version 5 data source.
5. Select **Data Sources** in the Additional Properties table.
6. Select the data source for which you want to configure connection pooling.
7. In the Additional Properties table, select **Connection Pool**. The properties to configure for a connection pool are shown in Figure 12-9:


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Home | Save | Preferences | Logout | Help |

Connection Pools

Connection pool properties that can be modified to change the behavior of the J2C connection pool manager. Default values are provided for non-production use. Reviewing and possible modification of these configuration values is recommended.

Configuration

General Properties

Scope	cells:demovm1Network:nodes:demovm1:servers:server1	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Connection Timeout	1800 seconds	Interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.
Max Connections	10 connections	The maximum number of ManagedConnections that can be created in this pool.
Min Connections	1 connections	The minimum number of ManagedConnections that should be maintained.
Reap Time	180 seconds	Interval, in seconds, between runs of the pool maintenance thread.
Unused Timeout	1800 seconds	Interval, in seconds, after which an unused connection is discarded by the pool maintenance thread.
Aged Timeout	0 seconds	Interval, in seconds, after which an unused, aged connection is discarded (regardless of recent usage activity) by the pool maintenance thread.
Purge Policy	EntirePool	Specifies how to purge connections when a "stale connection" or "fatal connection error" is detected.

Figure 12-9 Version 5 data source connection pool properties

– **Connection Timeout:** Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown. This can occur when the pool is at its maximum (**Max**

Connections) and all of the connections are in use by other applications for the duration of the wait.

For example, if Connection Timeout is set to `300` and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is not available within this time, the Pool Manager throws a *ConnectionWaitTimeoutException*.

Tip: If Connection Timeout is set to `0`, the Pool Manager waits as long as necessary until a connection is allocated.

– **Max Connections:** Specifies the maximum number of physical connections that can be created in this pool.

  These are the physical connections to the back-end database. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use is returned to the pool or a ConnectionWaitTimeoutException is thrown.

  For example, if Max Connections is set to `5`, and there are 5 physical connections in use, the Pool Manager waits for the amount of time specified in Connection Timeout for a physical connection to become free. If after that time there are still no free connections, the Pool Manager throws a ConnectionWaitTimeoutException to the application.

– **Min Connections:** Specifies the minimum number of physical connections to be maintained.

  Until this number is reached, the pool maintenence thread does not discard any physical connections. However, no attempt is made to bring the number of connections up to this number.

  For example if Min Connections is set to `3`, and one physical connection is created, that connection is not discarded by the **Unused Timeout** thread. By the same token, the thread does not automatically create two additional physical connections to reach the Min Connections setting.

– **Reap Time:** Specifies the interval, in seconds, between runs of the pool maintenance thread.

  For example, if Reap Time is set to `60`, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. When the pool maintenance thread runs it discards any connections that have been unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in Min Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

Tip: If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

- **Unused Timeout:** Specifies the interval in seconds after which an unused or idle connection is discarded.

  Tip: Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the Min Connections setting.

  For example, if the unused timeout value is set to `120`, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

- **Aged Timeout:** Specifies the interval in seconds before a physical connection is discarded, regardless of recent usage activity.

  Setting Aged Timeout to 0 allows active physical connections to remain in the pool indefinitely. For example if the Aged Timeout value is set to `1200`, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

  Tip: Set the Aged Timeout value higher than the Reap Timeout value for optimal performance.

- **Purge Policy**

  Specifies how to purge connections when a stale connection or fatal connection error is detected.

  Valid values are **EntirePool** and **FailingConnectionOnly**. If EntirePool is specified all physical connections in the pool are destroyed when a stale connection is detected. If FailingConnectionOnly is specified the pool attempts to destroy only the stale connection, the other connections remain in the pool. Final destruction of connections which are in use at the time of the error might be delayed. However, those connections are never returned to the pool.

8. Click **OK** when finished and save the configuration.

12.2.3 More information

These documents and Web sites are also relevant as further information sources:

- *WebSphere Connection Pooling* white paper

  `http://www.ibm.com/software/webservers/appserv/whitepapers.html`

- JDBC API documentation

  `http://java.sun.com/products/jdbc/index.html`

- JCA API documentation

  `http://java.sun.com/j2ee/connector/`

- EJB 2.0 documentation -persistence manager

  `http://java.sun.com/products/ejb/`

12.3 J2EE Connector Architecture (JCA) overview

The J2EE Connector architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EIS), for example, ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language. By defining a set of scalable, secure, and transactional mechanisms, the JCA enables the integration of EISs with application servers and enterprise applications.

JCA is part of J2EE 1.3, and as such, WebSphere Application Server V5 provides a complete implementation of the JCA 1.0 specification.

The **JCA Resource Adapter** is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the following functionality:

- Provides connectivity between J2EE components (an application server or an application client) and an EIS.
- Plugs into an application server.
- Collaborates with the application server to provide important services such as connection pooling, transaction and security services. JCA defines the

following set of system-level contracts between an application server and EIS:

  - A **connection management contract** that lets an application server pool connections to an underlying EIS, and lets application components connect to an EIS. This leads to a scalable application environment that can support a large number of clients requiring access to EISs.
  - A **transaction management contract** between the transaction manager and an EIS that supports transactional access to EIS resource managers. This contract lets an application server use a transaction manager to manage transactions across multiple resource managers. This contract also supports transactions that are managed internal to an EIS resource manager without the necessity of involving an external transaction manager.
  - A **security contract** that enables a secure access to an EIS. This contract provides support for a secure application environment, which reduces security threats to the EIS and protects valuable information resources managed by the EIS.

    The resource adapter implements the EIS-side of these system-level contracts.

- Implements the Common Client Interface (CCI) for EIS access. The CCI defines a standard client API through which a J2EE component accesses the EIS. This simplifies writing code to connect to an EIS's data store.

  The resource adapter provides connectivity between the EIS, the application server and the enterprise application via the CCI.

- Implements the standard Service Provider Interface (SPI) for integrating the transaction, security and connection management facilities of an application server (JCA Connection Manager) with those of a transactional resource manager.

A resource adapter is used within the address space of the application server and multiple resource adapters (that is, one resource adapter per type of EIS) are pluggable into an application server. This capability enables application components deployed on the application server to access the underlying EISs. This is shown in Figure 12-10 on page 576.

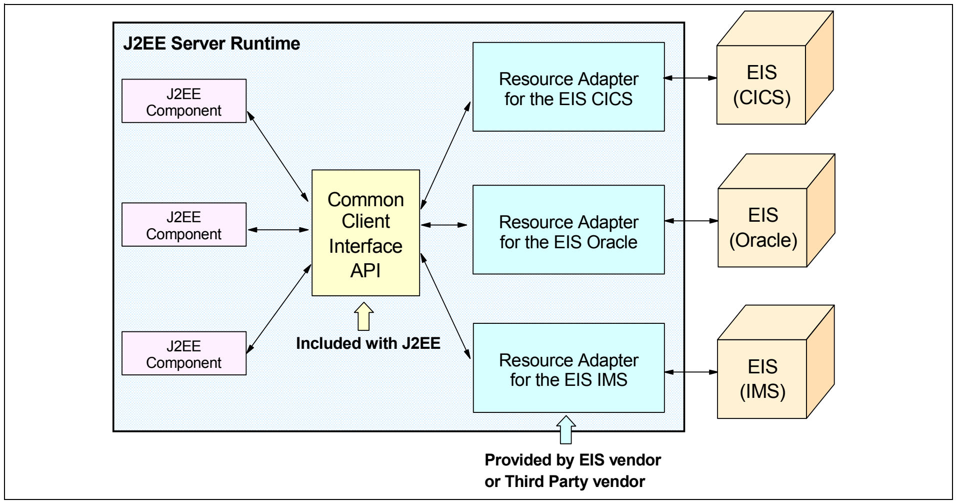

J2EE Server Runtime
J2EE Component
J2EE Component
J2EE Component
Common Client Interface API
Included with J2EE
Resource Adapter for the EIS CICS
Resource Adapter for the EIS Oracle
Resource Adapter for the EIS IMS
Provided by EIS vendor or Third Party vendor
EIS (CICS)
EIS (Oracle)
EIS (IMS)

Figure 12-10   Common Client Interface API

Benefits of JCA include:

- Once an application server implements JCA, any JCA-compliant resource adapter can plug in.
- Once a resource adapter implements JCA, it can plug in to any JCA compliant application server.
- Each EIS requires just one implementation of the resource adapter.
- The common client interface simplifies application integration with diverse EISs.

For more information on the J2EE Connector Architecture refer to the J2EE Connector Architecture documentation:

`http://java.sun.com/j2ee/connector/`

12.4 Using JCA resource adapters and connection factories with an application server

In WebSphere Version 5 two type of objects are configured for JCA support:

- Resource adapters
- Connection factories.

The role of the WebSphere administrator is to:

1. Install and define the resource adapter.
2. Define one or more connection factories associated with the resource adapter.

From the application point of view, the application using the resource adapter will request a connection from the connection factory through a JNDI lookup. The connection factory connects the application to the resource adapter.

12.4.1 Resource adapter

A WebSphere resource adapter administrative object represents the library that supplies implementation code for connecting applications to a specific EIS.

Resource adapters are stored in a Resource Adapter Archive (RAR) file, which is a Java archive (JAR) file used to package a resource adapter for the Connector Architecture. The file has a standard file extension .rar.

A RAR file can contain the following:

- EIS-supplied resource adapter implementation code in the form of JAR files or other executables, such as DLLs.
- Utility classes.
- Static documents, such as HTML files for developer documentation, not used for runtime.
- J2C common client interfaces, such as cci.jar.
- A mandatory deployment descriptor (ra.xml), which instructs the application server about how to use the resource adapter in an application server environment. The deployment descriptor contains information about the resource adapter, including security and transactional capabilities, and the ManagedConnectionFactory class name.

In WebSphere Application Server V5, RAR files are installed using the administrative console. However the RAR file or JCA resource adapter needs to be provided by your EIS vendor.

The only JCA resource adapter provided by WebSphere is the WebSphere Relational Resource Adapter, which is used to connect to relational databases using JDBC.

12.4.2 Connection factory

The WebSphere connection factory administrative object represents the configuration of a specific connection to the EIS supported by the resource adapter. The connection factory can be thought of as a holder of a list of connection configuration properties.

Application components, such as CMP 2.0 enterprise beans, have cmpConnectionFactory descriptors that refer to a specific connection factory, not to the resource adapter. In Example 12-4, the enterprise bean CustomerAccount is bound to the eis/webbankds_CMP connection factory.

Example 12-4 ibm-ejb-jar-bnd.xmi file for webbankEntityEJBs.jar

```
<ejbBindings
xmi:id="CustomerAccount_Bnd"jndiName="webbank/ejb/CustomerAccount">
    <enterpriseBean xmi:type="ejb:ContainerManagedEntity"
href="META-INF/ejb-jar.xml#CustomerAccount"/>
    <cmpConnectionFactory xmi:id="CMPConnectionFactoryBinding_2"
jndiName="eis/webbankds_CMP" resAuth="Per_Connection_Factory"/>
  </ejbBindings>
```

12.5 Using resource adapters

A resource adapter represents a library that supplies implementation code for connecting applications to a specific EIS back ends such as CICS or SAP. To use a resource adapter you need to install the resource adapter code and create connection factories that use the adapter.

One resource adapter, the WebSphere Relational Resource Adapter is predefined for you. Each time you check the **Use this DataSource in container managed persistence (CMP)** box when you create a data source a CMP connection factory is added to use this resource adapter.

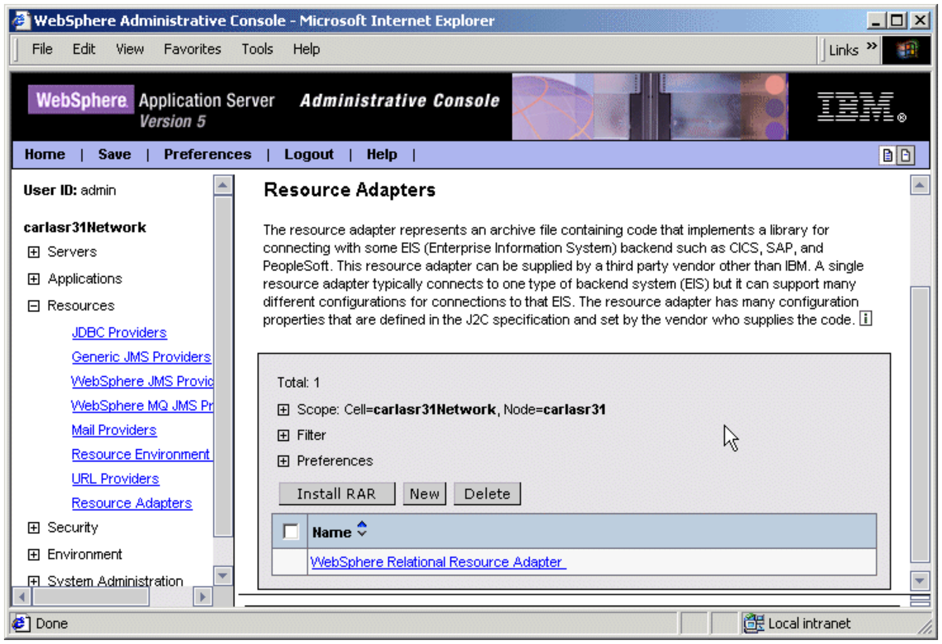
Resource adapter configuration is stored in the resources.xml file.

12.5.1 Installing a resource adapter

To install a resource adapter (.rar file) complete the following steps:

1. From the administrative console, expand **Resources** from the navigation tree.
2. Click **Resource Adapters**. The administrative console will show all the resource adapter objects configured. In Figure 12-11 on page 579 you see

one resource adapter configured, the WebSphere Relational Resource Adapter.


WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

User ID: admin

carlasr31Network
- Servers
- Applications
- Resources
  - JDBC Providers
  - Generic JMS Providers
  - WebSphere JMS Provid
  - WebSphere MQ JMS Pr
  - Mail Providers
  - Resource Environment
  - URL Providers
  - Resource Adapters
- Security
- Environment
- System Administration

Resource Adapters

The resource adapter represents an archive file containing code that implements a library for connecting with some EIS (Enterprise Information System) backend such as CICS, SAP, and PeopleSoft. This resource adapter can be supplied by a third party vendor other than IBM. A single resource adapter typically connects to one type of backend system (EIS) but it can support many different configurations for connections to that EIS. The resource adapter has many configuration properties that are defined in the J2C specification and set by the vendor who supplies the code.

Total: 1

Scope: Cell=**carlasr31Network**, Node=**carlasr31**

Filter

Preferences

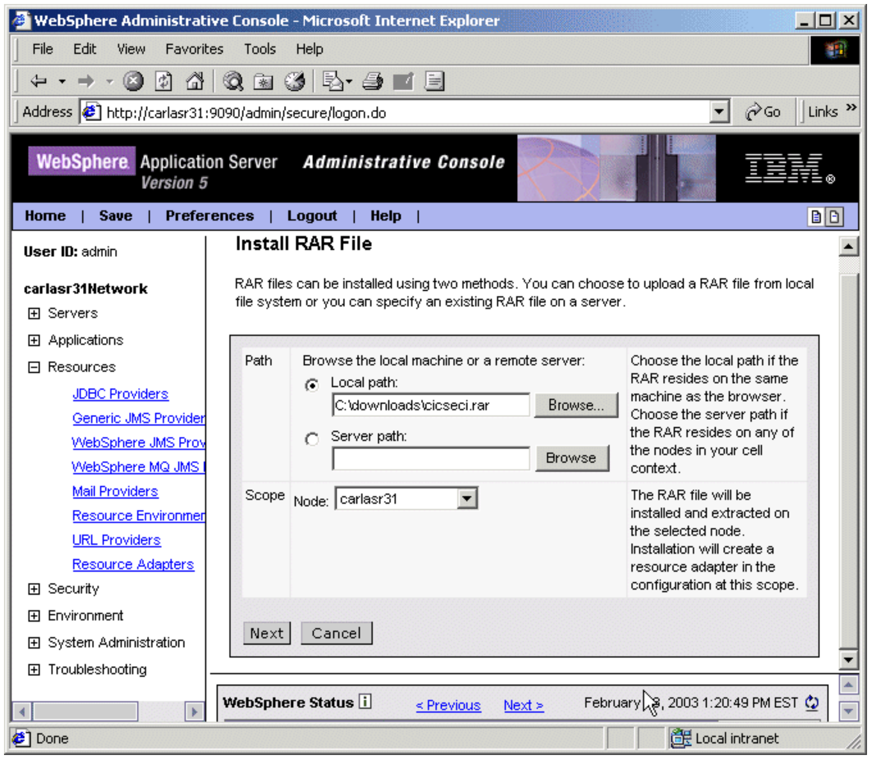
Install RAR | New | Delete

	Name
	WebSphere Relational Resource Adapter

Done | Local intranet

Figure 12-11 JCA resource adapters

3. Select **Install RAR** to install a new resource adapter.
4. Enter the path to the RAR file supplied by your EIS vendor. It can reside locally, on the same machine as the browser, or remotely, on any of the nodes in your cell. Note that WebSphere will install the RAR file to a different location.



Install RAR File

RAR files can be installed using two methods. You can choose to upload a RAR file from local file system or you can specify an existing RAR file on a server.

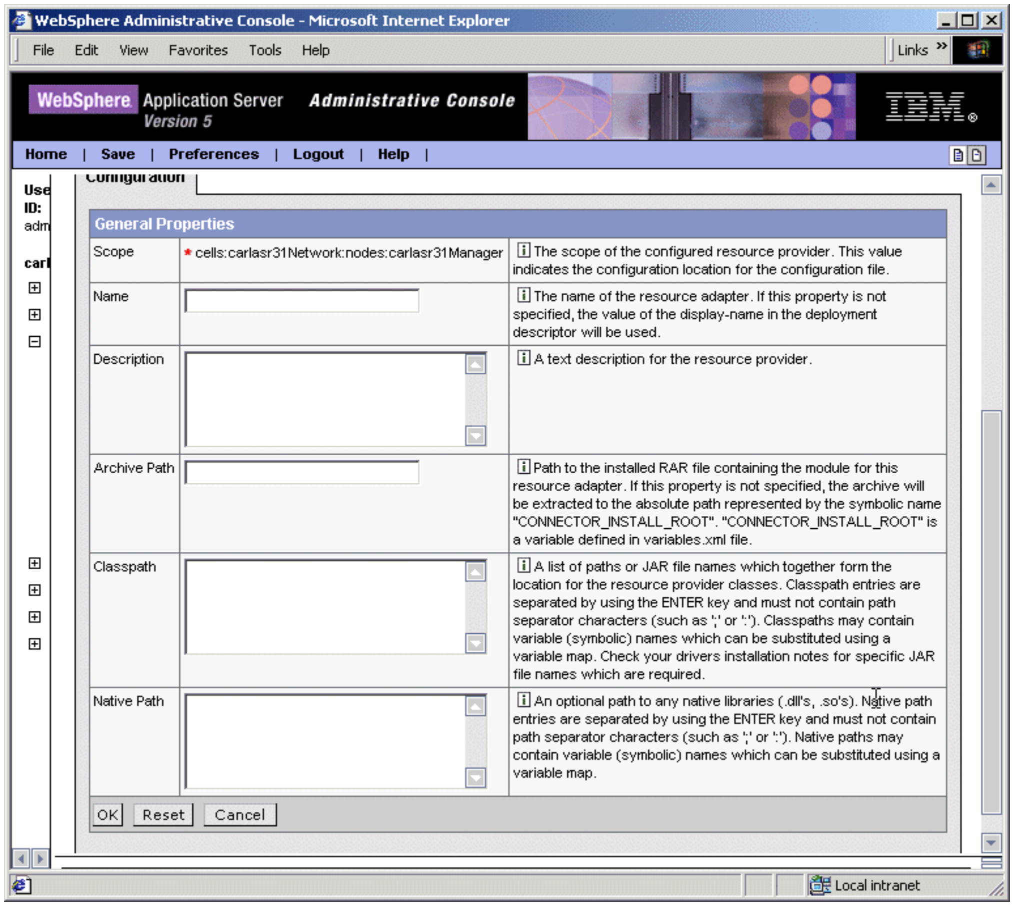
Path	Browse the local machine or a remote server: [✓ Local path:] C:\downloads\cicseci.rar Browse... Server path: Browse	Choose the local path if the RAR resides on the same machine as the browser. Choose the server path if the RAR resides on any of the nodes in your cell context.
Scope	Node: carlasr31	The RAR file will be installed and extracted on the selected node. Installation will create a resource adapter in the configuration at this scope.

February 3, 2003 1:20:49 PM EST

Figure 12-12 RAR file location

Select the node where you want to install the .rar file. Note that the .rar file is installed on a particular node. You will install the file on each node separately.

5. Click **Next**. The **Configuration** page for the resource adapter selected is displayed. This is shown in Figure 12-13 on page 581:


WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

WebSphere Application Server Version 5 Administrative Console

Home | Save | Preferences | Logout | Help |

General Properties

Scope	* cells:carlasr31Network:nodes:carlasr31Manager	The scope of the configured resource provider. This value indicates the configuration location for the configuration file.
Name		The name of the resource adapter. If this property is not specified, the value of the display-name in the deployment descriptor will be used.
Description		A text description for the resource provider.
Archive Path		Path to the installed RAR file containing the module for this resource adapter. If this property is not specified, the archive will be extracted to the absolute path represented by the symbolic name "CONNECTOR_INSTALL_ROOT". "CONNECTOR_INSTALL_ROOT" is a variable defined in variables.xml file.
Classpath		A list of paths or JAR file names which together form the location for the resource provider classes. Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.
Native Path		An optional path to any native libraries (.dll's, .so's). Native path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.

Figure 12-13 JCA resource adapter properties

In this example you do not have to configure any properties. The defaults combined with the information supplied in the .rar file provide all the information needed. However, you have the option of configuring the following:

- **Name**: An administrative name for the resource adapter.
- **Description**: An optional description of the resource adapter, for your administrative records.
- **Archive path**: The path where the Resource Adapter Archive (RAR) file is going to be installed. If this property is not specified, the archive will be extracted to the absolute path represented by the ${CONNECTOR_INSTALL_ROOT} variable. The default is <WAS_HOME>/installedConnectors/<adaptername.rar>

Note: The resource adapter is only installed for the node selected in the Install window, and only the resources.xml file for that node gets updated. The relational resource adapter provided with WebSphere is the only one configured at the cell level, and is installed under <WAS_HOME>/lib.

 – **Classpath**: A list of paths or JAR file names which together form the location for the resource adapter classes. The resource adapter codebase itself (.rar file) is automatically added to the classpath.
 – **Native path**: A list of paths which together form the location for the resource adapter native libraries (.dll's, .so's)

6. Click **OK**.

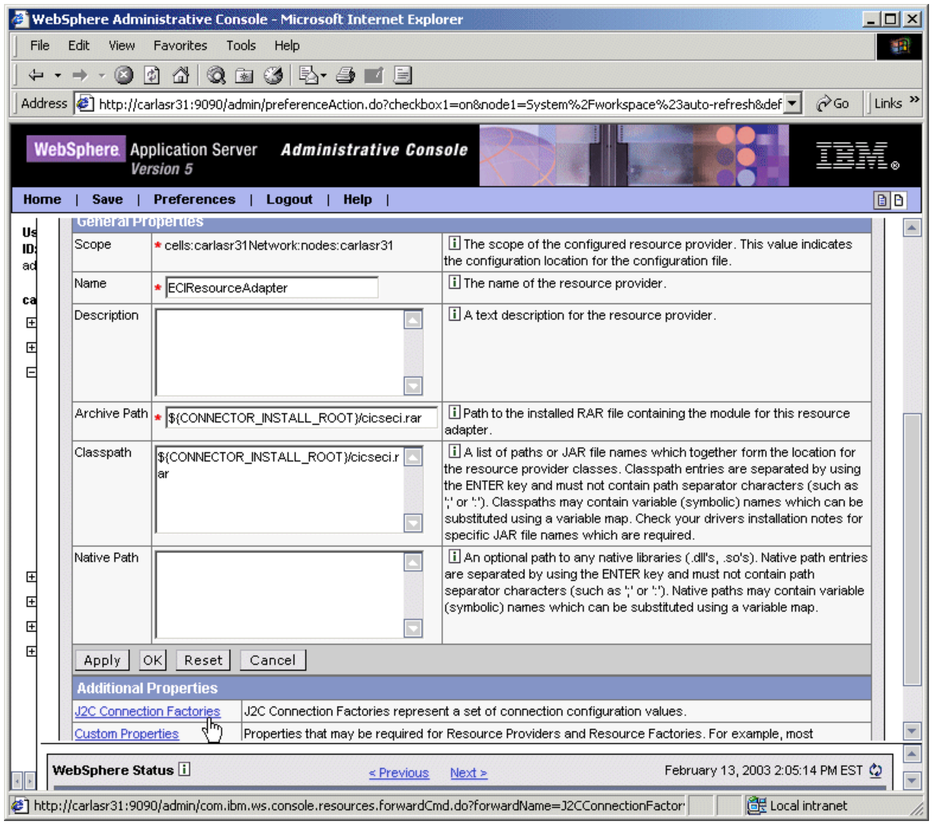
12.5.2 Configuring J2C connection factories

A J2C connection factory represents a set of connection configuration values. Application components such as EJBs have <resource-ref> descriptors that refer to the connection factory, not the resource adapter. The connection factory is really just a holder of a list of connection configuration properties. In addition to the arbitrary set of configuration properties defined by the vendor of the resource adapter, there are several standard configuration properties that apply to the connection factory. These standard properties are used by the connection pool manager in the application server runtime and are not used by the vendor supplied resource adapter code.

To create a J2C connection factory complete the following steps:

1. Select the J2C resource adapter just created. You will now have entries in the Additional Properties table, allowing you access to connection factories, custom properties, and the deployment descriptor (ra.xml).
2. Click **J2C Connection Factories** in the Additional Properties table as shown in Figure 12-14 on page 583:

Note: The terms J2C and JCA both refer to J2EE Connector Architecture and they are used here interchangeably.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/preferenceAction.do?checkbox1=on&node1=System%2Fworkspace%23auto-refresh&def
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
General Properties

Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource provider. This value indicates the configuration location for the configuration file.
Name	* ECIResourceAdapter	The name of the resource provider.
Description		A text description for the resource provider.
Archive Path	* ${CONNECTOR_INSTALL_ROOT}/cicseci.rar	Path to the installed RAR file containing the module for this resource adapter.
Classpath	${CONNECTOR_INSTALL_ROOT}/cicseci.rar	A list of paths or JAR file names which together form the location for the resource provider classes. Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.
Native Path		An optional path to any native libraries (.dll's, .so's). Native path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.

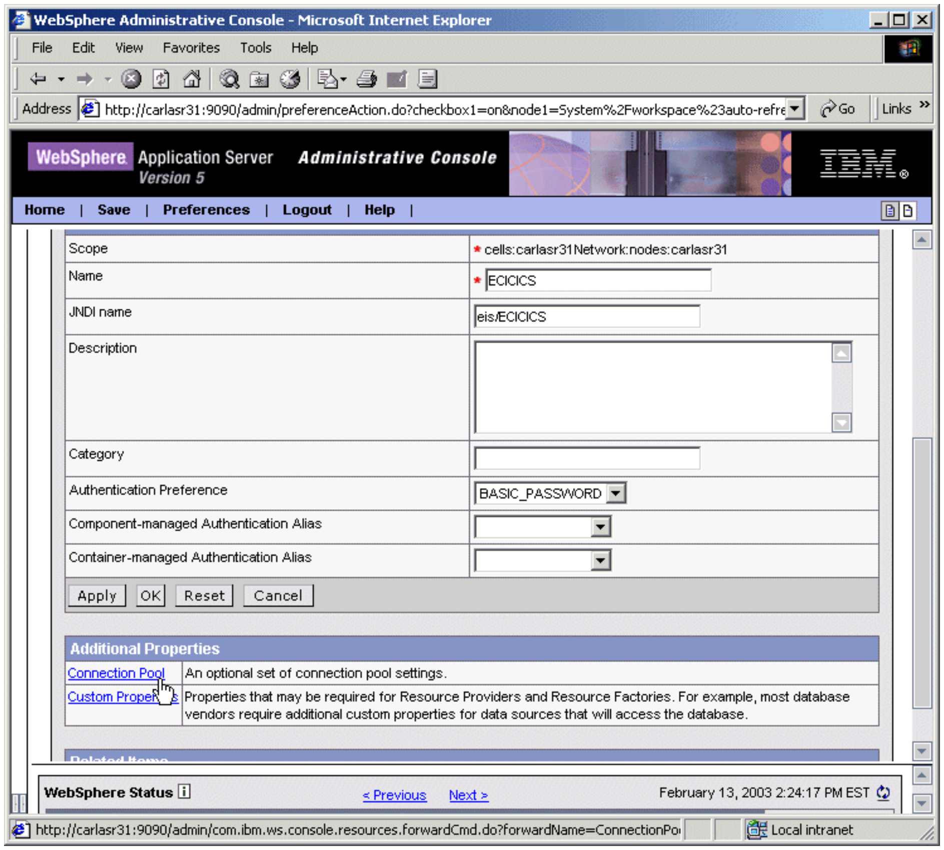
J2C Connection Factories	J2C Connection Factories represent a set of connection configuration values.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most

WebSphere Status
< Previous Next >
February 13, 2003 2:05:14 PM EST
http://carlasr31:9090/admin/com.ibm.ws.console.resources.forwardCmd.do?forwardName=J2CConnectionFactor
Local intranet

Figure 12-14 Configurable resources for the JCA connector

3. Click **New** to create a new connection factory, or select an existing one to modify the connection factory properties.

   The J2C Connection Factory Configuration page is shown in Figure 12-15 on page 584.


WebSphere Administrative Console - Microsoft Internet Explorer

Address: http://carlasr31:9090/admin/preferenceAction.do?checkbox1=on&node1=System%2Fworkspace%23auto-refre

WebSphere Application Server Version 5 — Administrative Console

Home | Save | Preferences | Logout | Help |

Scope	* cells:carlasr31Network:nodes:carlasr31
Name	* ECICICS
JNDI name	eis/ECICICS
Description	
Category	
Authentication Preference	BASIC_PASSWORD
Component-managed Authentication Alias	
Container-managed Authentication Alias	

Apply | OK | Reset | Cancel

Additional Properties

Connection Pool	An optional set of connection pool settings.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

WebSphere Status < Previous Next > February 13, 2003 2:24:17 PM EST

http://carlasr31:9090/admin/com.ibm.ws.console.resources.forwardCmd.do?forwardName=ConnectionPo — Local intranet

Figure 12-15 J2C connection factory properties

The general properties are:

– **Name**. An administrative name for the J2C connection factory

– **JNDI name**. The connection factory name to be registered in the application server's name space, including any naming subcontext.

  When installing an application that contains modules with J2C resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resource.

Tip: As a convention, use the value of the Name property prefixed with "eis/" (such as "eis/ConnectionFactoryName").

– **Description**. An optional description of the J2C connection factory, for your administrative records.

- **Category**. Specifies a category that you can use to classify or group the connection factory.
- **Authentication mechanism preference**. Specifies which of the authentication mechanisms that are defined for the corresponding resource adapter applies to this connector factory.

  For example, if two authentication mechanism entries have been defined for a resource adapter (KerbV5 and Basic Password), this will specify one of those two types. If the authentication mechanism preference specified is not an authentication mechanism available on the corresponding resource adapter, it is ignored.

  Note: The authentication mechanisms defined for a resource adapter can be seen in the resource adapter descriptor file (ra.xml). For example:

  ```
  <authentication-mechanism>
  <description>BasicPassword authentication</description>

  <authentication-mechanism-type>BasicPassword</authentication-mechanism
  -type>

  <credential-interface>javax.resource.spi.security.PasswordCredential</
  credential-interface>
    </authentication-mechanism>

  - <authentication-mechanism>
    <description>Kerbv5 Authentication</description>

  <authentication-mechanism-type>Kerbv5</authentication-mechanism-type>

  <credential-interface>javax.resource.spi.security.GenericCredential</c
  redential-interface>
    </authentication-mechanism>
  ```

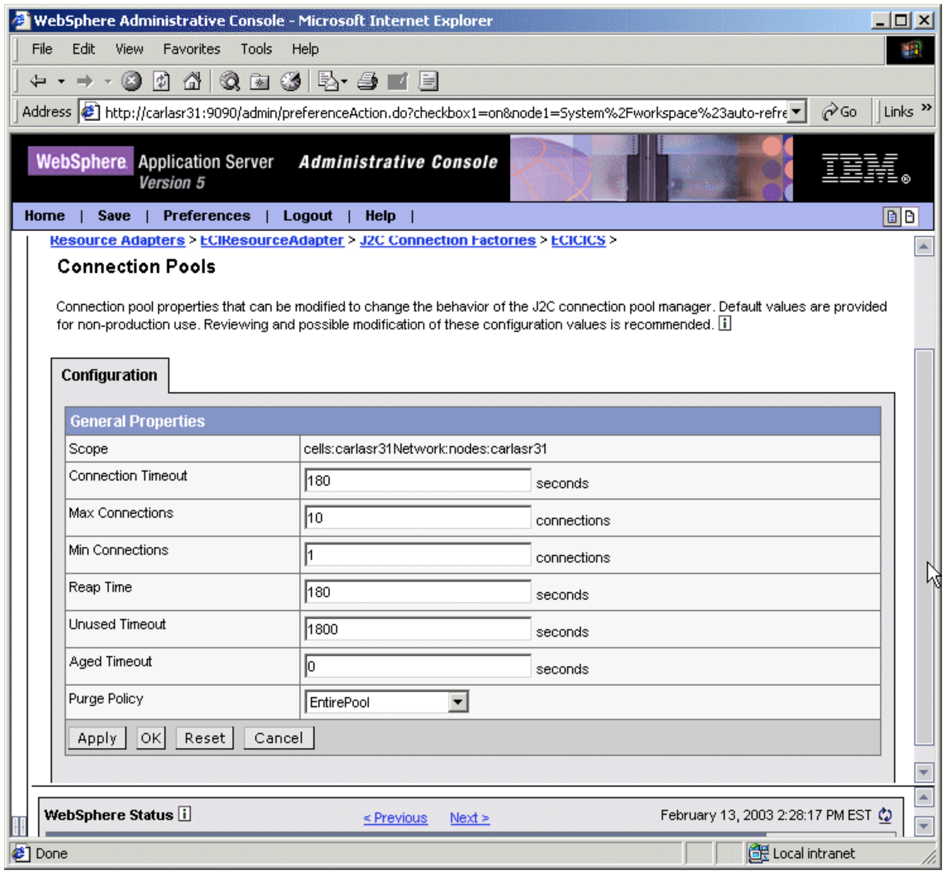
- **Container-managed authentication alias**. The authentication alias used for container-managed sign-on to the resource.
- **Component-managed authentication alias**. The authentication alias used for component-managed sign-on to the resource.

4. Click **OK** to complete the configuration or **Apply** to continue with further configuration for custom properties or connection pooling. Before you click **Apply** you will not see the Additional Properties table with these options.
5. When done, click **Save**, to save the configuration.

Configuring connection pooling for a J2C connection factory

Connection pool properties affect the behavior of the J2C connection manager. Though default values are provided, it is recommended that you review these settings so ensure they are appropriate to your production environment.

Connection pool configuration parameters are accessed from the Additional Properties table in the J2C connection factory configuration page (see Figure 12-15 on page 584). The properties to configure for a J2C connection factory connection pool are as shown in Figure 12-16.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/preferenceAction.do?checkbox1=on&node1=System%2Fworkspace%23auto-refre
Go Links
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
Resource Adapters > ECIResourceAdapter > J2C Connection Factories > ECICICS >

Connection Pools

Connection pool properties that can be modified to change the behavior of the J2C connection pool manager. Default values are provided for non-production use. Reviewing and possible modification of these configuration values is recommended.

Configuration

General Properties		
Scope	cells:carlasr31Network:nodes:carlasr31	
Connection Timeout	180	seconds
Max Connections	10	connections
Min Connections	1	connections
Reap Time	180	seconds
Unused Timeout	1800	seconds
Aged Timeout	0	seconds
Purge Policy	EntirePool	

Figure 12-16 J2C connection factory connection pool properties

– **Connection Timeout:** Specifies the interval, in seconds, after which a connection request times out. This can occur when the pool is at its maximum (**Max Connections**) and all of the connections are in use by other applications for the duration of the wait.

For example, if Connection Timeout is set to `300` and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is not available within this time, the Pool Manager throws a ConnectionWaitTimeoutException.

> **Tip:** If Connection Timeout is set to `0`, the Pool Manager waits as long as necessary until a connection is allocated.

– **Max Connections:** Specifies the maximum number of ManagedConnections that can be created in this pool.

  These are the physical connections to the back-end database. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use is returned to the pool or a ConnectionWaitTimeoutException is thrown.

  For example, if Max Connections is set to `5`, and there are 5 physical connections in use, the Pool Manager waits for the amount of time specified in Connection Timeout for a physical connection to become free. If after that time there are still no free connections, the Pool Manager throws a ConnectionWaitTimeoutException to the application.

– **Min Connections:** Specifies the minimum number of ManagedConnections (physical connections) to be maintained.

  Until this number is reached, the pool maintenance thread does not discard any physical connections. However, no attempt is made to bring the number of connections up to this number.

  For example if Min Connections is set to `3`, and one physical connection is created, that connection is not discarded by the **Unused Timeout** thread. By the same token, the thread does not automatically create two additional physical connections to reach the Min Connections setting.

– **Reap Time:** Specifies the interval, in seconds, between runs of the pool maintenance thread.

  For example, if Reap Time is set to `60`, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. When the pool maintenance thread runs it discards any connections that have been unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in Min Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

Tip: If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

– **Unused Timeout:** Specifies the interval in seconds after which an unused or idle connection is discarded.

  Tip: Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the Min Connections setting.

  For example, if the unused timeout value is set to `120`, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value.

– **Aged Timeout:** Specifies the interval in seconds before a physical connection is discarded, regardless of recent usage activity.

  Setting Aged Timeout to 0 allows active physical connections to remain in the pool indefinitely. For example if the Aged Timeout value is set to `1200`, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, is affected by the Reap Time value. See Reap Time for more information.

  Tip: Set the Aged Timeout value higher than the Reap Timeout value for optimal performance.

– **Purge Policy:** Specifies how to purge connections when a stale connection or fatal connection error is detected.

  Valid values are **EntirePool** and **FailingConnectionOnly**. If EntirePool is specified all physical connections in the pool are destroyed when a stale connection is detected. If FailingConnectionOnly is specified the pool attempts to destroy only the stale connection, the other connections remain in the pool. Final destruction of connections which are in use at the time of the error might be delayed. However, those connections are never returned to the pool.

6. Click **OK** when finished and save the configuration.

12.5.3 Using resource adapters from an application

The following code example shows how you might access the CICS ECI resource adapter we just installed from an application. This code snippet assumes you have a resource reference called “eis/ref/ECICICS” that points to a javax.resource.cci.ConnectionFactory with JNDI name “eis/ECICICS”. It's a minimal sample, with no connection factory caching, etc.

Example 12-5 Using resource adapters from an application. Code sample

```
private int getRate(String source) throws java.lang.Exception {

   // get JNDI context
   javax.naming.InitialContext ctx = new javax.naming.InitialContext();
   // get local JNDI environment
   javax.naming.Context env =
(javax.naming.Context)ctx.lookup("java:comp/env");
   javax.resource.cci.ConnectionFactory connectionFactory connectionFactory =
      (javax.resource.cci.ConnectionFactory) env.lookup("eis/ref/ECICICS");

   // get a connection to the EIS
   javax.resource.cci.Connection connection =
connectionFactory.getConnection();

   // create an interaction and a CICS ECI specific interaction spec
   javax.resource.cci.Interaction interaction =
connection.createInteraction();
   com.ibm.connector2.cics.ECIInteractionSpec interactionSpec = new
com.ibm.connector2.cics.ECIInteractionSpec();

   // create the comm area record
   source = (source.trim().toUpperCase()+"            ").substring(0, 12);
   GenericRecord record = new GenericRecord((source).getBytes("IBM037"));

   // set the CICS program name we want to call
   interactionSpec.setFunctionName("CALCRATE");

   // invoke the CICS program
   interaction.execute(interactionSpec, record, record);

   // close the interation and the connection
   interaction.close();
   connection.close();
   // get the results from the return comm area record
   byte[] commarea = record.getCommarea();
   int value = Integer.parseInt(new String(commarea,
"IBM037").substring(8,12).trim());
   return value;

}
```

12.5.4 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter. Search on JCA.

  `http://www.ibm.com/software/webservers/appserv/infocenter.html`

- J2EE Connector Architecture documentation

  `http://java.sun.com/j2ee/connector`

12.6 JavaMail and JavaMail service providers

The JavaMail APIs provide a platform and protocol-independent framework for building Java-based mail client applications.

The JavaMail APIs are generic for sending and receiving mail. They require service providers, known in WebSphere as protocol providers, to interact with mail servers that run the pertaining protocols.

A JavaMail provider encapsulates a collection of protocol providers. WebSphere Application Server V5 has a Built-in Mail Provider that encompasses three protocol providers: SMTP, IMAP and POP3. These protocol providers are installed as the default and should be sufficient for most applications.

1. **Simple Mail Transfer Protocol (SMTP)**. This is a popular transport protocol for sending mail. JavaMail applications can connect to an SMTP server and send mail through it by using this SMTP protocol provider.
2. **Post Office Protocol (POP3)**. This is the standard protocol for receiving mail.
3. **Internet Message Access Protocol (IMAP)**. This is an alternative protocol to POP3 for receiving mail.

   To use other protocols, you must install the appropriate service provider for those protocols.

Note: In this section the terms JavaMail provider and mail provider are used interchangeably.

In addition to service providers, JavaMail requires the Java Activation Framework (JAF) as the underlying framework to deal with complex data types that are not plain text, like Multipurpose Internet Mail Extensions (MIME), Uniform Resource Locator (URL) pages, and file attachments.

The JavaMail APIs, the JAF, the service providers and the protocols are shipped as part of WebSphere Application Server V5 using the following Sun licensed packages:

- **mail.jar**: Contains the JavaMail APIs, and the SMTP, IMAP, and POP3 service providers.
- **activation.jar**: Contains the JavaBeans Activation Framework.

These JAR files can be found in <WAS_HOME>/java/jre/lib/ext directory along with all the other Java extension packages provided with WebSphere.

Figure 12-17 illustrates the relationship among the different JavaMail components.

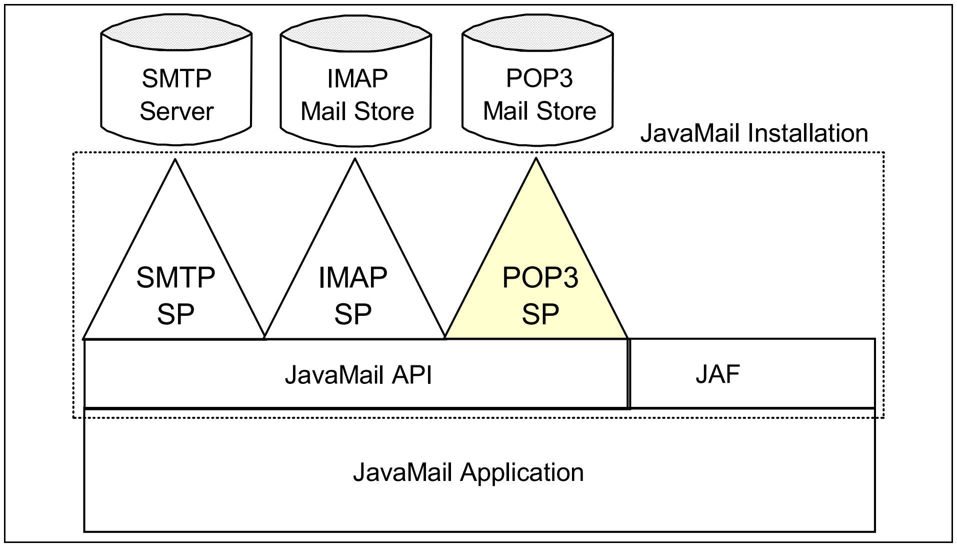


Figure 12-17 JavaMail components

WebSphere Application Server V5 supports JavaMail Version 1.2 and the JavaBeans Activation Framework (JAF) Version 1.0. All Web components of WebSphere (servlets, JSPs, EJBs, and application clients), support JavaMail.

12.7 JavaMail sessions

A JavaMail session object (or session administrative object) is a resource used by the application to obtain connections to a mail server. A mail session object manages the configuration options and user authentication information used to interact with the mail system. Refer to the *JavaMail API Design Specification, Version 1.2* document for more information on the Session class:

```
http://java.sun.com/products/javamail/JavaMail-1.2.pdf
```

JavaMail sessions are configured to use a particular JavaMail provider.

12.7.1 Configuring the mail provider

A mail provider encapsulates a collection of protocol providers. Protocol providers interact with JavaMail APIs and mail servers running those protocols. WebSphere Application Server V5 has a built-in mail provider that encompasses three protocol providers: SMTP, IMAP and POP3. These protocol providers are installed by default and should be sufficient for most applications. However you can configure a new provider if necessary.

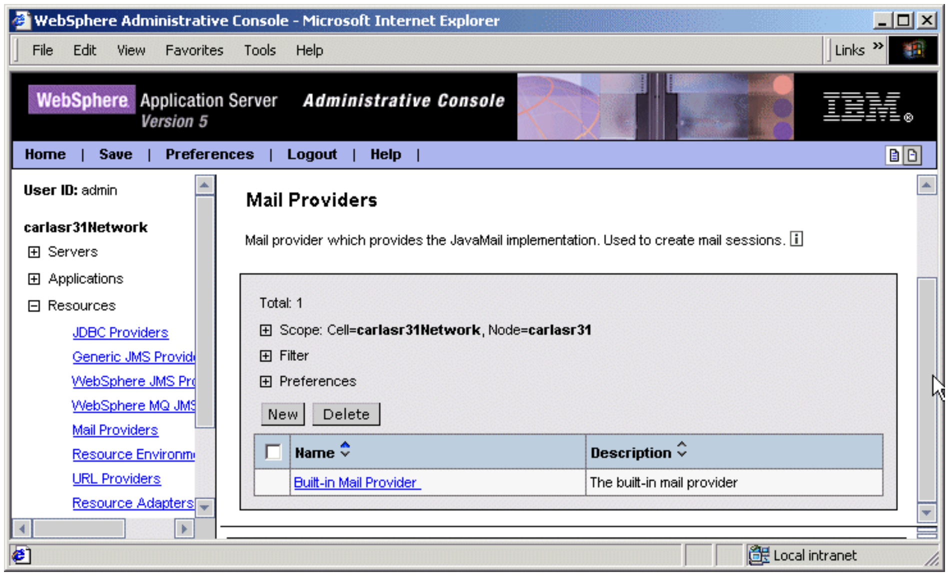
Refer to the *JavaMail API Design Specification, Version 1.2* document for more information on how service providers (mail providers) can register their protocol implementations so they can be used by JavaMail APIs:

```
http://java.sun.com/products/javamail/JavaMail-1.2.pdf
```

To configure a new mail provider with WebSphere complete the following steps from the administrative console:

1. Expand **Resources** from the navigation tree.
2. Click **Mail Providers**.
3. Select **Scope** and click **Apply**. The scope determines whether JavaMail resources configured to use this provider will be available at the cell, node or the application server level.

   Figure 12-18 on page 593, shows the mail provider installed with WebSphere. The Built-in Mail Provider is configured at the cell level, meaning resources configured for this provider will be available to all the application servers in the cell.



Mail Providers

Mail provider which provides the JavaMail implementation. Used to create mail sessions.

Total: 1

Scope: Cell=carlasr31Network, Node=carlasr31

Filter

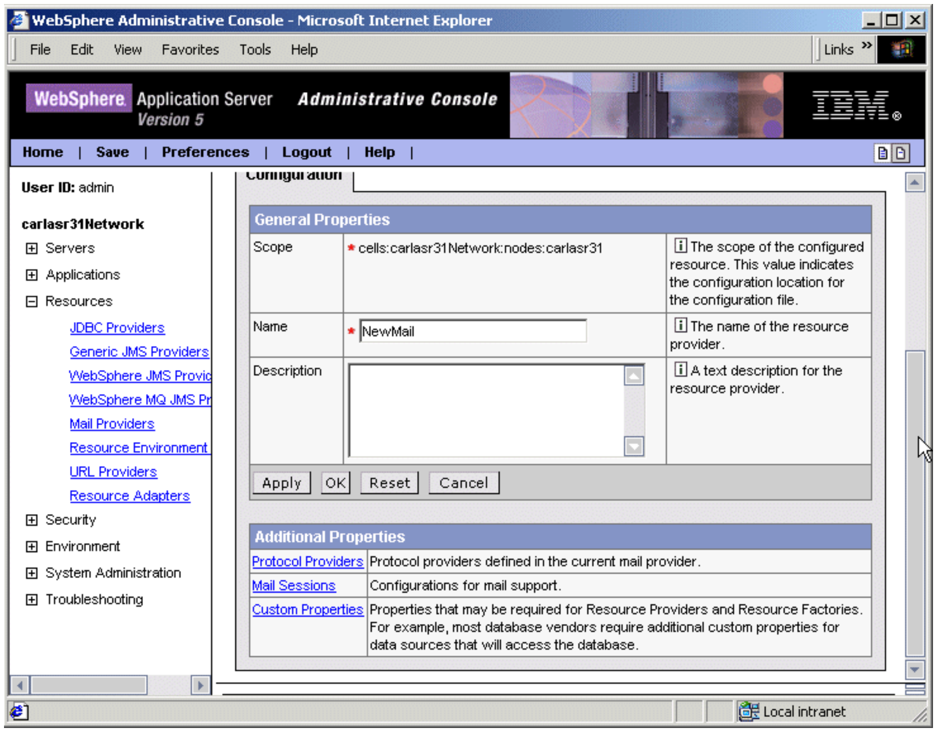
Preferences

New Delete

	Name	Description
	Built-in Mail Provider	The built-in mail provider

Figure 12-18 Mail provider page

4. Click **New** to configure a new mail provider. Enter a name and description and click **Apply**. The properties required to configure a new mail provider are shown in Figure 12-19 on page 594.

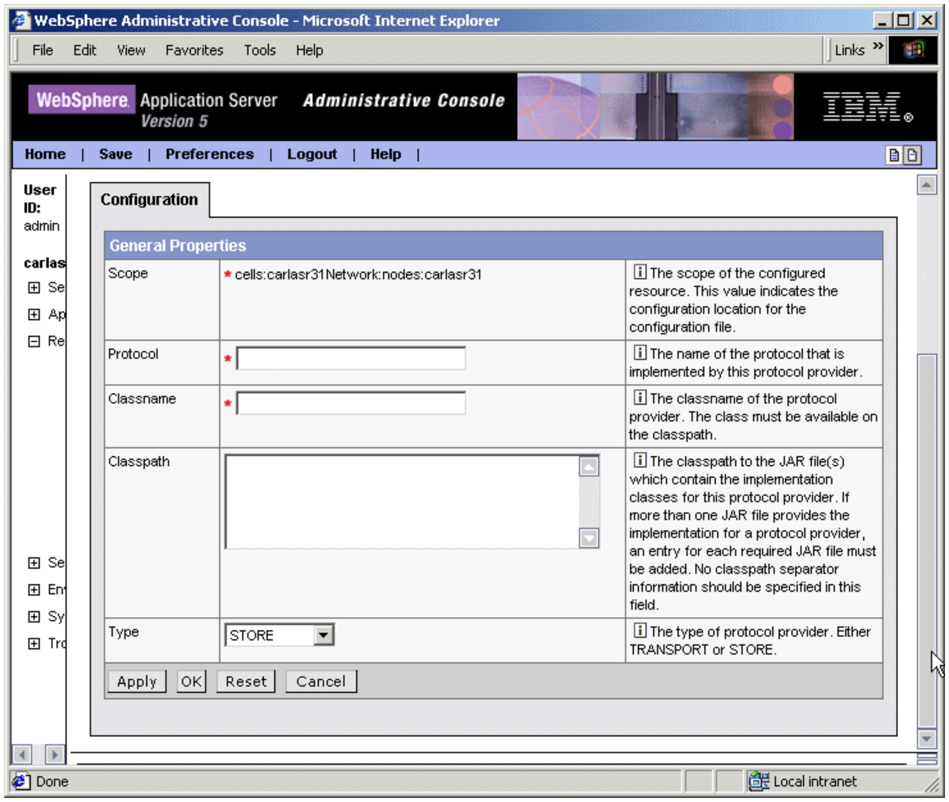


General Properties		
Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* NewMail	The name of the resource provider.
Description		A text description for the resource provider.

Additional Properties	
Protocol Providers	Protocol providers defined in the current mail provider.
Mail Sessions	Configurations for mail support.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Figure 12-19 Mail Provider general properties

5. Click **Protocol Providers** in the Additional Properties table.
6. Click **New** to add a protocol provider.


WebSphere Administrative Console - Microsoft Internet Explorer

Configuration

General Properties

Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Protocol	*	The name of the protocol that is implemented by this protocol provider.
Classname	*	The classname of the protocol provider. The class must be available on the classpath.
Classpath		The classpath to the JAR file(s) which contain the implementation classes for this protocol provider. If more than one JAR file provides the implementation for a protocol provider, an entry for each required JAR file must be added. No classpath separator information should be specified in this field.
Type	STORE	The type of protocol provider. Either TRANSPORT or STORE.

File Edit View Favorites Tools Help
Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
carlas
Se
Ap
Re
Se
En
Sy
Tr
Apply
OK
Reset
Cancel
Done
Local intranet

Figure 12-20 Protocol provider configuration page

The properties to configure are:

– **Protocol**: Specifies the protocol name.

– **Classname**: Specifies the implementation class for the specific protocol provider. The class must be available in the classpath.

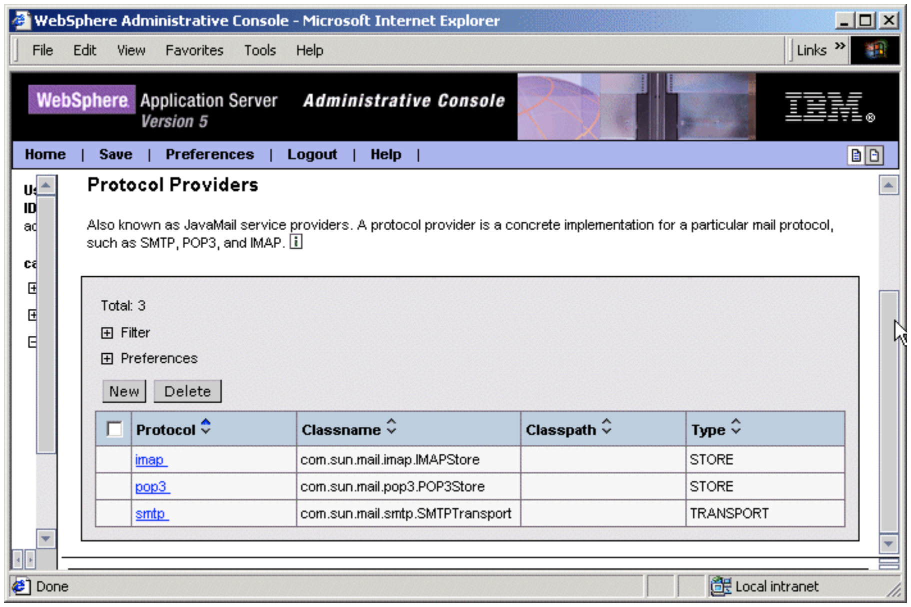
– **Classpath**: Specifies the path to the JAR files that contain the implementation classes for this protocol provider.

– **Type**: Specifies the type of protocol provider. Valid options are:

  • STORE. The protocol is used for receiving mail (Message Access Protocol).

  • TRANSPORT. The protocol is used for sending mail (Transport Protocol).

Figure 12-21 on page 596 shows the protocol providers implemented with the built-in mail provider.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
Done
Local intranet

Protocol Providers

Also known as JavaMail service providers. A protocol provider is a concrete implementation for a particular mail protocol, such as SMTP, POP3, and IMAP.

Total: 3

Filter

Preferences

New Delete

	Protocol	Classname	Classpath	Type
	imap	com.sun.mail.imap.IMAPStore		STORE
	pop3	com.sun.mail.pop3.POP3Store		STORE
	smtp	com.sun.mail.smtp.SMTPTransport		TRANSPORT

Figure 12-21 Protocol providers

Refer to the JavaMail API documentation for more information on the above protocol providers:

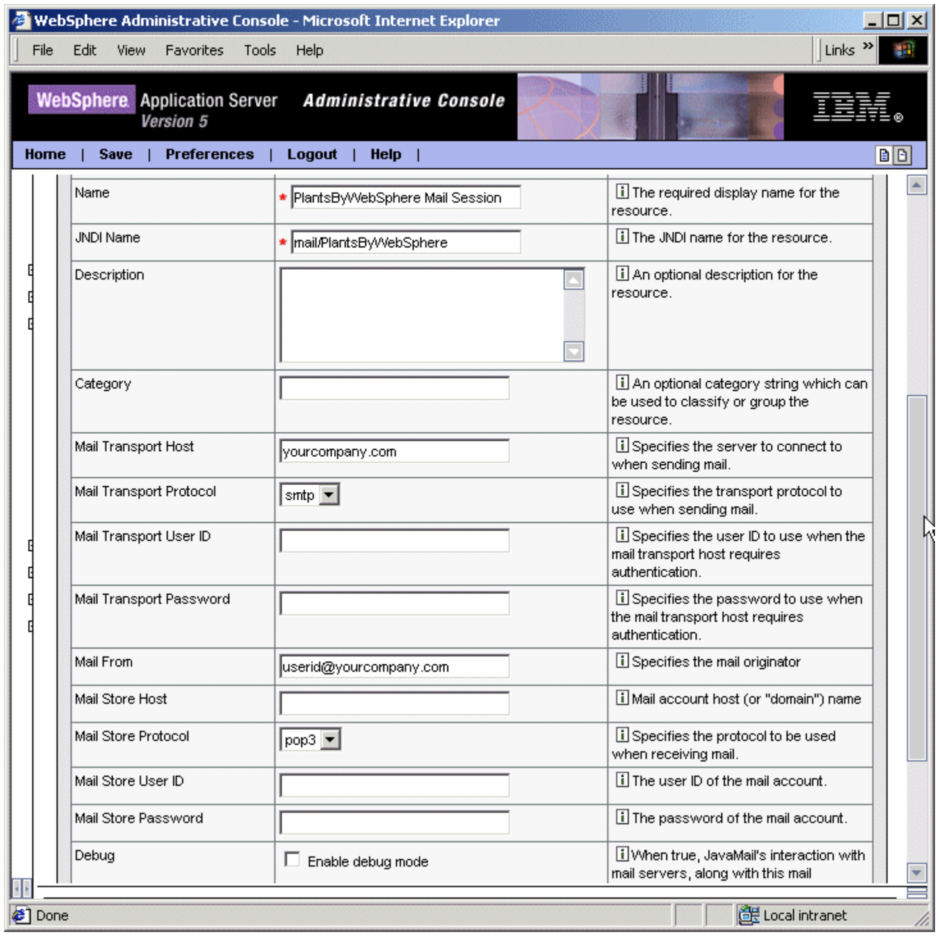
`http://java.sun.com/products/javamail/1.3/docs/javadocs/`

7. Click **OK**.
8. Click **Save**, to save the configuration.

12.7.2 Configuring JavaMail sessions

To configure JavaMail sessions with a particular mail provider complete the following steps from the administrative console:

1. Expand **Resources** from the navigation tree.
2. Click **Mail Providers**.
3. Select **Scope** and click **Apply**.
4. Select the mail provider used by the JavaMail session.
5. Select **Mail Sessions** in the Additional Properties table (see Figure 12-19 on page 594).
6. Select **New** to create a new mail session object, or select an existing one to update it. Figure 12-22 on page 597 shows the configuration page for the PlantsByWebSphere sample application.



WebSphere Administrative Console - Microsoft Internet Explorer

Name	* PlantsByWebSphere Mail Session	The required display name for the resource.
JNDI Name	* mail/PlantsByWebSphere	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Mail Transport Host	yourcompany.com	Specifies the server to connect to when sending mail.
Mail Transport Protocol	smtp	Specifies the transport protocol to use when sending mail.
Mail Transport User ID		Specifies the user ID to use when the mail transport host requires authentication.
Mail Transport Password		Specifies the password to use when the mail transport host requires authentication.
Mail From	userid@yourcompany.com	Specifies the mail originator
Mail Store Host		Mail account host (or "domain") name
Mail Store Protocol	pop3	Specifies the protocol to be used when receiving mail.
Mail Store User ID		The user ID of the mail account.
Mail Store Password		The password of the mail account.
Debug	☐ Enable debug mode	When true, JavaMail's interaction with mail servers, along with this mail

Figure 12-22 Configuration page for the mail session

The properties to note are:

– **Name**: The administrative name of the JavaMail session object.

– **JNDI Name**: The JavaMail session object name as registered in the application servers name space, including any naming subcontext.

When installing an application that contains modules with JavaMail resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

Tip: As a convention, use the value of the Name property prefixed with “mail/” (such as “mail/mailsessionName”).

- **Mail Transport Host**: Specifies the server to connect to when sending mail. Specify the fully qualified Internet host name of the mail server.
- **Mail Transport Protocol**: Specifies the transport protocol to use when sending mail. It could be SMTP or any transport protocol for which the user has installed a provider.
- **Mail Transport userid**: Specifies the user ID to provide when connecting to the mail transport host. This setting is not generally used for most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.
- **Mail Transport password**: Specifies the password to provide when connecting to the mail transport host. Like the user ID, this setting is rarely used by most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.
- **Mail From**: Specifies the mail originator. This value represents the Internet e-mail address that, by default, displays in the received message, as either the “From” or the “Reply-To” address. The recipient's reply comes to this address
- **Mail Store Host**: Specifies the server to which to connect when receiving mail. This setting combines with the mail store user ID and password to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com®, then the mail store host is itso.ibm.com.

d. **Mail Store Protocol**: Specifies the protocol to be used when receiving mail. It could be IMAP, POP3 or any store protocol for which the user has installed a provider.

e. **Mail Store userid**: Specifies the user ID to use when connecting to the mail store. This setting combines with the mail store host and password to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com then the user ID is *itso*.

f. **Mail Store password**: Specifies the password to use when connecting to the mail store host. This property combines with the mail store user ID and host to represent a valid mail account. For example, if the mail account is itso@itso.ibm.com then enter the password for user ID *itso*.

g. **Debug**: Toggles debug mode on and off for this mail session. When true, JavaMail’s interaction with mail servers, along with this mail session’s properties will be printed to <stdout>.

7. Click **OK** and save the configuration.

12.7.3 Example code

The code segment shown in Example 12-6 illustrates how an application component sends a message and saves it to the "Sent" folder.

Example 12-6 JavaMail application code

```
//get JavaMail session

javax.naming.InitialContext ctx = new javax.naming.InitialContext();
javax.mail.Session mail_session = (javax.mail.Session)
ctx.lookup("java:comp/env/mail/MailSession");

      //prepare message

       MimeMessage msg = new MimeMessage(mail_session);
       msg.setRecipients(Message.RecipientType.TO,
InternetAddress.parse("bob@coldmail.net"));
msg.setFrom(new InternetAddress("alice@mail.eedge.com"));
       msg.setSubject("Important message from eEdge.com");
       msg.setText(msg_text);

      //send message

      Transport.send(msg);

      //save message in "Sent" folder

      Store store = mail_session.getStore();
      store.connect();
     Folder f = store.getFolder("Sent");
     if (!f.exists()) f.create(Folder.HOLDS_MESSAGES);
     f.appendMessages(new Message[] {msg});
```

Refer to the JavaMail API Design Specification, Version 1.2 document for other examples:

http://java.sun.com/products/javamail/JavaMail-1.2.pdf

12.7.4 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter. Search on "JavaMail".

  http://www.ibm.com/software/webservers/appserv/infocenter.html

- JavaMail API design specification

  `http://java.sun.com/products/javamail/JavaMail-1.2.pdf`

12.8 URL providers

A Uniform Resource Locator (URL) is an identifier that points to a resource that is accessible electronically, such as a directory on a network machine, a file in a directory or a document stored in a database.

URLs are in the format scheme:scheme_information:

- **Scheme**. A scheme might be http, ftp, file, or another term that identifies the mechanism or protocol by which the resource can be accessed.

  In a Web browser location or address box, a URL for a file available using HyperText Transfer Protocol (HTTP) starts with “http:”. An example is: http://www.ibm.com/software/webservers/appserv/infocenter.html

  Files available using File Transfer Protocol (FTP) start with “ftp:”. Files available locally start with “file:”.

- **Scheme_information**. The scheme_information commonly identifies the Internet machine making a resource available, the path to that resource, and the resource name.

  The scheme_information for HTTP, FTP and File generally starts with two slashes (//), then provides the Internet address separated from the resource path name with one slash (/). For example:

  ```
  http://www.ibm.com/software/webservers/appserv/library.html
  ```

  Note: For HTTP and FTP, the path name ends in a slash when the URL points to a directory. In such cases, the server generally returns the default index for the directory.

  ```
  For example, the URL:
  http://www.ibm.com/software/webservers/appserv/library.html
  ```

- Has the scheme: `http`
- Has the scheme_information:
  `//www.ibm.com/software/webservers/appserv/library.html`

From the scheme information we can identify the machine where the information resides (`www.ibm.com`), the path (`/software/webservers/appserv/`), and the resource name (`library.html`).

Note: A URL can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80:

`http://w3.ibm.com:80/` is equivalent to `http://w3.ibm.com/`

A URL provider implements the functionality for a particular URL protocol, such as HTTP, by extending the java.net.URLStreamHandler and java.net.URLConnection classes. It enables communication between the application and a URL resource that is served by that particular protocol.

A URL provider named Default URL Provider is included in the initial WebSphere configuration. This provider utilizes the URL support provided by the IBM JDK. Any URL resource with protocols based on the Java 2 Standard Edition 1.3.1, such as HTTP, FTP or File, can use the default URL provider.

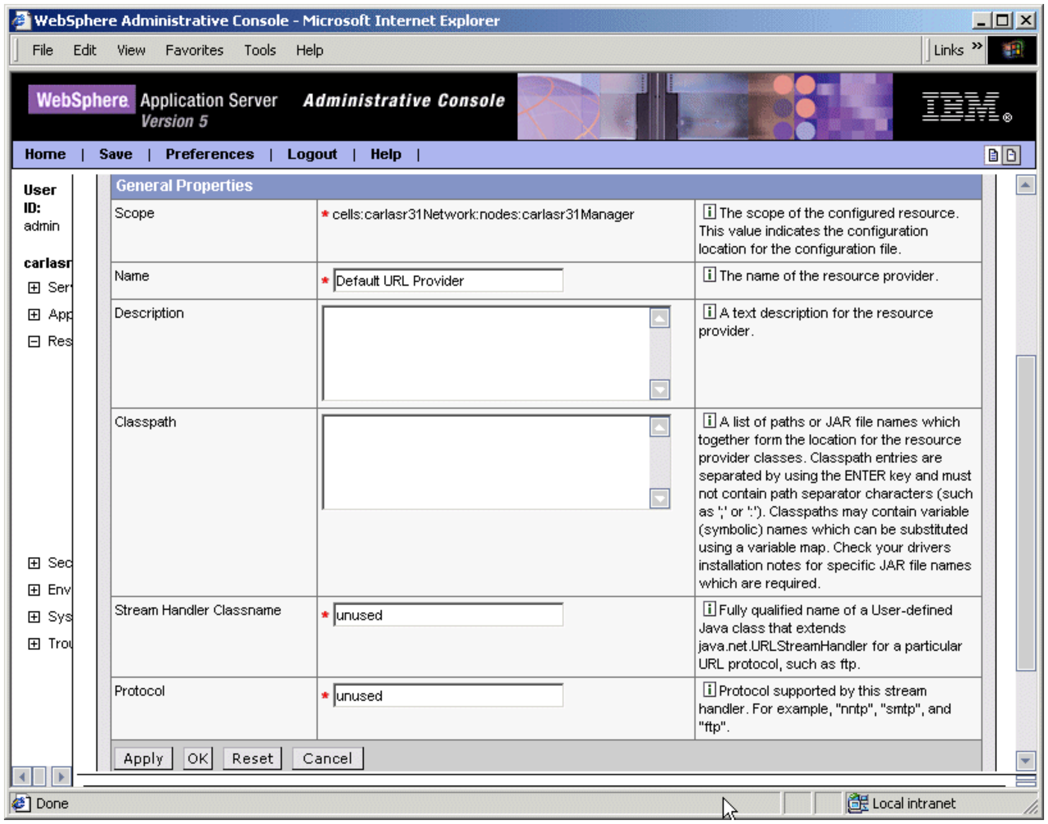
Customers can also "plug in" their own URL providers that implement other protocols not supported by the JDK.

12.8.1 Configuring URL providers and URLs

URL resource objects are administrative objects used by an application to communicate with an URL. These resource objects are used to read from an URL or to write to an URL. URL resource objects use URL providers for class implementation.

To configure or create an URL provider from the administrative console, complete the following tasks:

1. Expand **Resources** from the navigation tree.
2. Click **URL Providers**.
3. Select **Scope** and click **Apply**.
4. Click **New** to configure a new URL provider, or select an existing one to edit it.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr
Ser
App
Res
Sec
Env
Sys
Trou
Apply OK Reset Cancel
Done
Local intranet

General Properties		
Scope	* cells:carlasr31Network:nodes:carlasr31Manager	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* Default URL Provider	The name of the resource provider.
Description		A text description for the resource provider.
Classpath		A list of paths or JAR file names which together form the location for the resource provider classes. Classpath entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.
Stream Handler Classname	* unused	Fully qualified name of a User-defined Java class that extends java.net.URLStreamHandler for a particular URL protocol, such as ftp.
Protocol	* unused	Protocol supported by this stream handler. For example, "nntp", "smtp", and "ftp".

Figure 12-23 URL provider configuration page

The properties to configure are:

– **Name**. An administrative name for the URL provider.

– **Description**. An optional description of the URL provider, for your administrative records.

– **Classpath**. A list of paths or JAR file names which together form the location for the URL provider classes.

– **Stream Handler Class Name**. Specifies the fully qualified name of the Java class that implements the stream handler for the protocol specified by the Protocol property. A stream protocol handler knows how to make a connection for a particular protocol type, such as HTTP or FTP. It extends the java.net.URLStreamHandler class for that particular protocol.

– **Protocol**. Specifies the protocol supported by this stream handler, for example “http” or “ftp”.

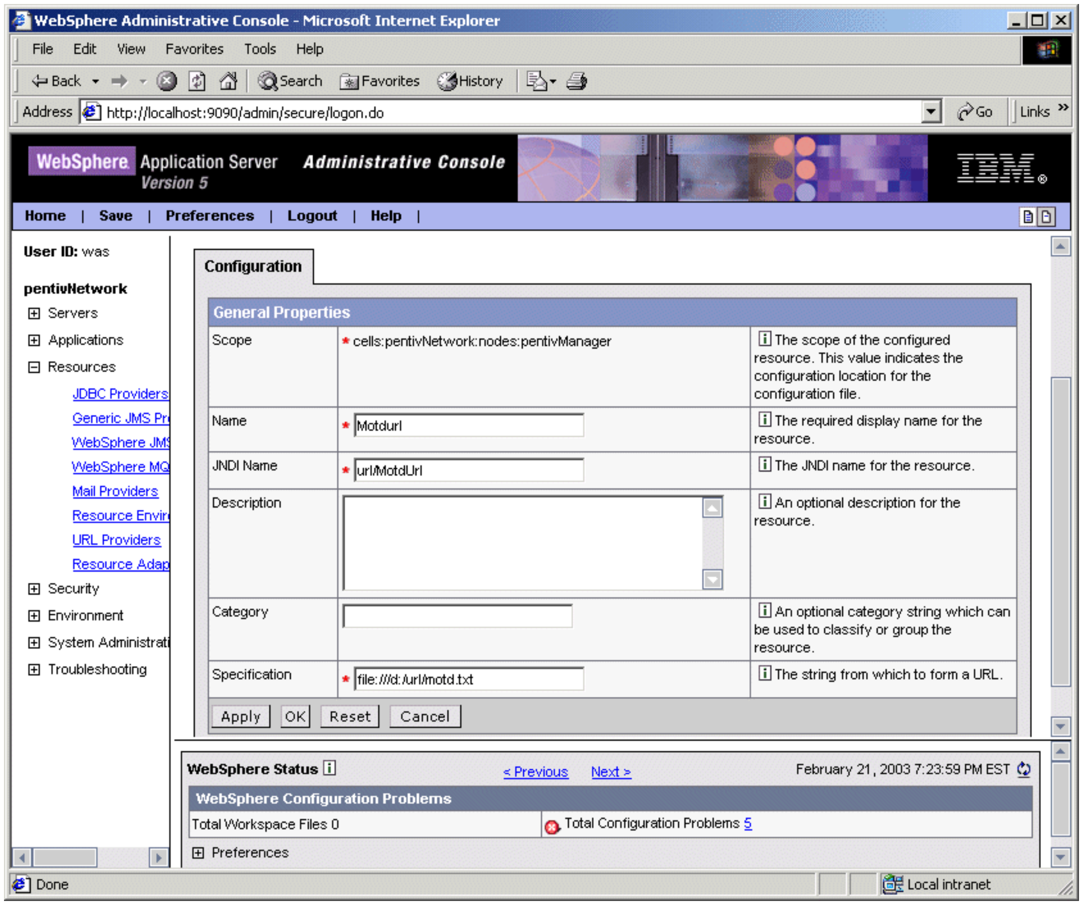
5. Click **OK** and save the configuration.

Important: You need to manually install the URL provider (a set of JARs) on each node where the URL provider is going to be used and ensure that it is included in the classpath above.

12.8.2 Configuring URLs

To configure an URL administrative object that points to an electronically accessible resource, such as a directory or a file on a network machine, complete the following steps from the administrative console:

1. Expand **Resources** from the navigation tree.
2. Click **URL Providers**.
3. Select **Scope** and click **Apply**.
4. Select the URL provider that implements the protocol required to access the URL resource.
5. Select **URLs** in the Additional Properties table. Select **New**:



WebSphere Administrative Console - Microsoft Internet Explorer

Address: http://localhost:9090/admin/secure/logon.do

User ID: was

pentivNetwork

Configuration

General Properties

Scope	* cells:pentivNetwork:nodes:pentivManager	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* Motdurl	The required display name for the resource.
JNDI Name	* url/MotdUrl	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Specification	* file:///d:/url/motd.txt	The string from which to form a URL.

WebSphere Status — February 21, 2003 7:23:59 PM EST

WebSphere Configuration Problems

Total Workspace Files 0	Total Configuration Problems 5

Figure 12-24 Defining URLs

The properties to configure are:

– **Name**: Specifies the administrative name for the URL resource object.

– **JNDI Name**: The URL session object name as registered in the application servers name space, including any naming subcontext.

  When installing an application that contains modules with URL resource references, the resources defined by the deployment descriptor of the module need to be bound to the actual JNDI name of the resources.

> **Tip:** As a convention, use the value of the Name property prefixed with “url/” (such as “url/UrlName”).

- **Description**. An optional description of the URL object, for your administrative records.
- **Category**. Specifies a category that you can use to classify or group the URL object.
- **Spec**. Specifies the URL resource to which this URL object is bound, for example “`file:///d:/url/motd.txt`”.

6. Click **OK** and save the configuration.

12.8.3 URL provider sample

Example 12-7 provides a code sample making use of the URL provider and URL resources. Note that the Web module resource reference, myHttpUrl, is bound to the URL resource JNDI name, url/MotdUrl, during application assembly or deployment.

Example 12-7 HTTP URL provider sample

```
javax.naming.InitialContext ctx = new javax.naming.InitialContext();
javax.naming.Context env =
   (javax.naming.Context) ctx.lookup("java:comp/env");
java.net.URL url = (java.net.URL) env.lookup("myHttpUrl");
java.io.InputStream ins = url.openStream();
int c;
while ((c = ins.read()) != -1) {
   out.write(c);
   }
```

In this case, we inserted the Example 12-7 code into a JSP, added the JSP to a Web module, added a URL resource reference to the Web module, and deployed the Web module. Then we checked that the contents of the file specified in the MotdUrl URL resource, "file:///d:/url/motd.txt", were included in the JSP’s output.

Similarly, a stock quote custom URL provider could be accessed as shown in Example 12-8. The Web module resource reference, myQuoteUrl, is bound to a URL resource with JNDI name, url/QuoteUrl, and URL “quote://IBM”. The custom URL provider will access an online stock quote for IBM.

Example 12-8 Quote URL provider sample

```
javax.naming.InitialContext ctx = new javax.naming.InitialContext();
javax.naming.Context env =
   (javax.naming.Context) ctx.lookup("java:comp/env");
java.net.URL url = (java.net.URL) env.lookup("myQuoteUrl");
out.println("The stock price is "+url.getContent());
```

Note: Each application server's name space is initialized on startup. This means application servers must be restarted to load a modified resource property, such as a URL string.

12.8.4 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter. Search on URL.

  http://www.ibm.com/software/webservers/appserv/infocenter.html
- URL class documentation.

  http://java.sun.com/j2se/1.3/docs/api/
- More information on the types of URLs and their formats can be found at:

  http://java.sun.com/products/jdk/1.2/docs/api/java/net/URL.html

12.9 Resource environment providers

The java:comp/env environment provides a single mechanism by which both JNDI name space objects and local application environment objects can be looked up. WebSphere Application Server provides a number of local environment entries by default.

The J2EE 1.3 specification also provides a mechanism for defining custom (non-default) environment entries using <resource-env-ref> entries defined in an application's standard deployment descriptors. The J2EE 1.3 specification separates the definition of the resource environment entry from the application by:

1. Requiring the application server to provide a mechanism for defining separate administrative objects that encapsulate a resource environment entry. The administrative objects are to be accessible via JNDI in the application server's local name space (java:comp/env). The J2EE 1.3 specification does not specify how an application server should provide this functionality. As a result, the mechanism is generally application server product specific.
2. Specifying the administrative object's JNDI lookup name and the expected returned object type in the <resource-env-ref>.

As an example, Example 12-9 shows a resource environment entry defined in an application's Web module deployment descriptor (web.xml). Example 12-10 shows how this resource environment entry could be accessed from Java code in the Web module.

Example 12-9 Resource-env-ref in deployment descriptor

```
<web-app>
.....
<resource-env-ref>
   <resource-env-ref-name>myapp/MyLogWriter</resource-env-ref-name>
   <resource-env-ref-type>com.ibm.itso.test.LogWriter</resource-env-ref-type>
</resource-env-ref>
.....
</web-app>
```

Example 12-10 Java code to access resource environment reference

```
import com.ibm.itso.test.*;
.....
InitialContext ctx = new InitialContext();
LogWriter myLog = (LogWriter) ctx.lookup("java:comp/env/myapp/MyLogWriter");
myLog.write(msg);
.....
```

12.9.1 WebSphere V5 support for resource environment references

WebSphere Application Server supports the <resource-env-ref> mechanism by providing resource environment provider (and associated objects) administrative objects that are configured using the administration tools. Each <resource-env-ref> requires the creation of the following administered objects in the order shown:

1. Resource environment provider

   Defines an administrative object that groups together the referenceable, resource env entry administrative objects and any required custom properties.

   The provider can be defined at the cell, node or application server scope. The scope chosen determines which resources.xml configuration file is updated to contain the provider’s configuration stanza:

   ```
   <resources.env:ResourceEnvironmentProvider
   xmi:id="ResourceEnvironmentProvider_1" name="ResProviderName"/>
   ```

2. Referenceable

   Defines the classname of the factory class that returns object instances implementing a Java interface.

The referenceable's configuration is added to the provider's stanza in the resources.xml file appropriate to the scope:

```
<resources.env:ResourceEnvironmentProvider
xmi:id="ResourceEnvironmentProvider_1" name="ResProviderName">
    <referenceables xmi:id="Referenceable_1"
factoryClassname="com.ibm.itso.test.LogWriterFactory"
classname="com.ibm.itso.test.LogWriter"/>
  </resources.env:ResourceEnvironmentProvider>
```

3. Resource env entry

   Defines the binding target (JNDI name), factory class and return object type (via link to the Referenceable) of the resource environment entry.

   The referenceable's configuration is added to the provider's stanza in the *resources.xml* file appropriate to the scope:

```
<resources.env:ResourceEnvironmentProvider
xmi:id="ResourceEnvironmentProvider_1" name="ResProviderName">
   <factories xmi:type="resources.env:ResourceEnvEntry"
xmi:id="ResourceEnvEntry_1" name="MyLogWriter" jndiName="myapp/MyLogWriter"
referenceable="Referenceable_1"/>
   <referenceables xmi:id="Referenceable_1"
factoryClassname="com.ibm.itso.test.LogWriterFactory"
classname="com.ibm.itso.test.LogWriter"/>
  </resources.env:ResourceEnvironmentProvider>
```

As shown in Table 12-2 on page 608, the scope (cell, node or application server) at which the resource environment provider, and associated objects, are created will determine whether the resource environment entry is available when a particular application server is started.

Table 12-2 Scope versus resource environment entry availability

Scope	XML configuration file	Details
Cell	cells/<cellname>/resources.xml	All application servers in the cell will have the resource environment entry defined in their local runtime.
Node	cells/<cellname>/nodes/<nodename>/ resources.xml	All application servers in the node will have the resource environment entry defined in their local runtime. Application servers on other nodes in the cell will not have this resource environment entry.
Server	cells/<cellname>/nodes/<nodename>/ servers/<servername>/resources.xml	Only the specific application server will have the resource environment entry in its runtime.

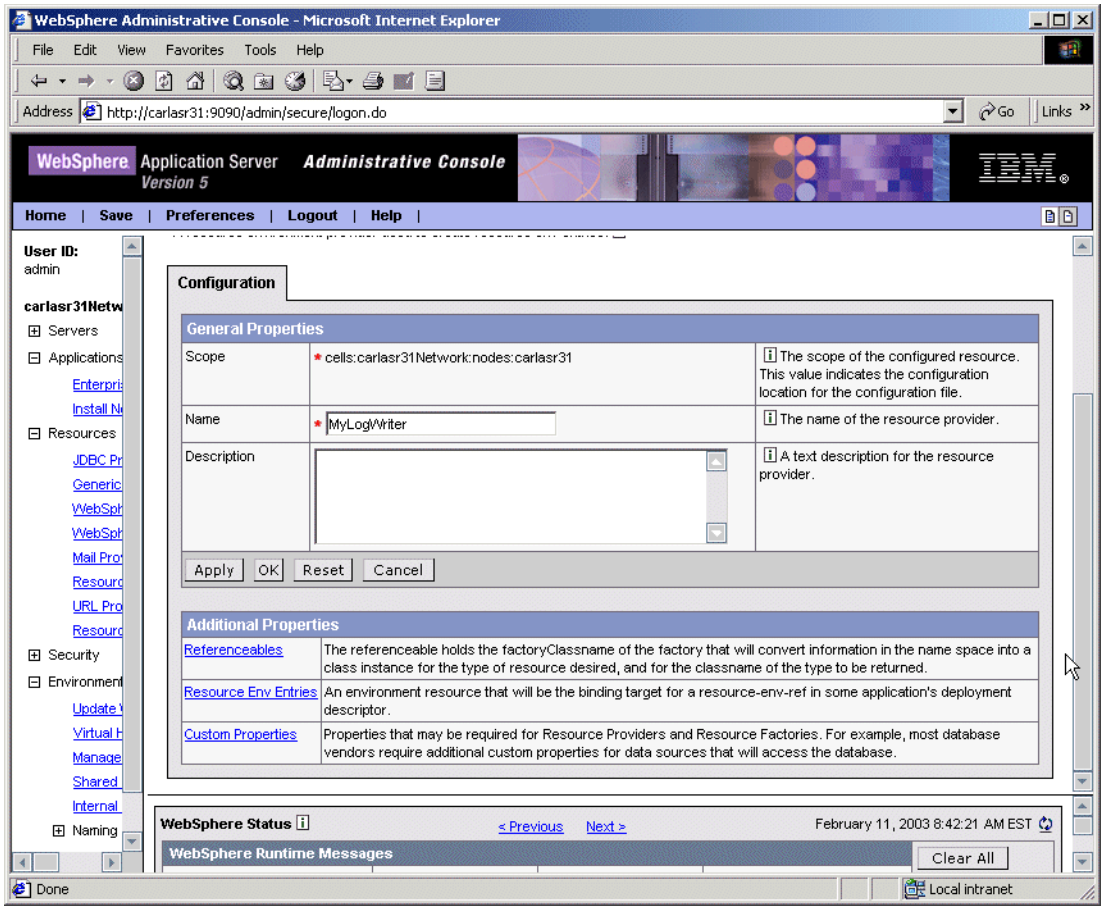
For example, when an application server is started, it reads its runtime configuration (including resource environment entries) from configuration files at the cell, node and server level. As a result, if the resource environment entry administered objects are defined in the cell-level resources.xml, the definitions will be read by all application servers on startup. If defined in the server-level resources.xml, only that application server will read the definition and load the environment entry into its runtime.

Details on how to create and configure each of the above administrative objects necessary to support a resource environment reference (<resource-env-ref>) are found in 12.9.2, “Configuring the resource environment provider” on page 609.

12.9.2 Configuring the resource environment provider

To create settings for a resource environment provider:

1. Click **Resources -> Manage Resource Environment Provider** in the navigation tree.
2. Click **New**.
3. Enter a name and description for the new resource environment provider.
4. Click **Apply**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Netw
Servers
Applications
Enterpri
Install N
Resources
JDBC Pr
Generic
WebSph
WebSph
Mail Pro
Resourc
URL Pro
Resourc
Security
Environment
Update
Virtual H
Manage
Shared
Internal
Naming
Configuration
Apply OK Reset Cancel
WebSphere Status
< Previous Next >
February 11, 2003 8:42:21 AM EST
WebSphere Runtime Messages
Clear All
Done
Local intranet

General Properties

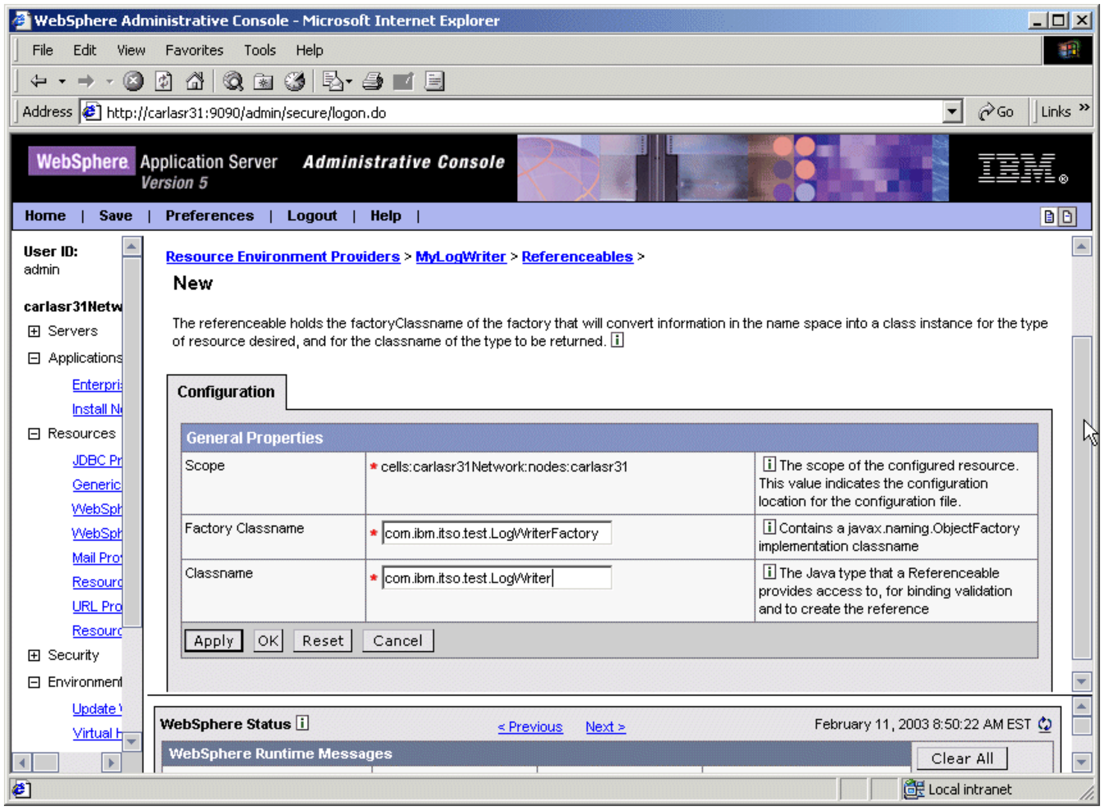
Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* MyLogWriter	The name of the resource provider.
Description		A text description for the resource provider.

Additional Properties

Referenceables	The referenceable holds the factoryClassname of the factory that will convert information in the name space into a class instance for the type of resource desired, and for the classname of the type to be returned.
Resource Env Entries	An environment resource that will be the binding target for a resource-env-ref in some application's deployment descriptor.
Custom Properties	Properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

Figure 12-25 Creating a resource environment provider

5. Click **Referenceables** in the Additional Properties table.
6. Click **New**. Use this page to set the classname of the factory that will convert information in the name space into a class instance for the type of resource desired.


Resource Environment Providers > MyLogWriter > Referenceables >

New

The referenceable holds the factoryClassname of the factory that will convert information in the name space into a class instance for the type of resource desired, and for the classname of the type to be returned.

Configuration

General Properties

Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Factory Classname	* com.ibm.itso.test.LogWriterFactory	Contains a javax.naming.ObjectFactory implementation classname
Classname	* com.ibm.itso.test.LogWriter	The Java type that a Referenceable provides access to, for binding validation and to create the reference

WebSphere Administrative Console - Microsoft Internet Explorer
http://carlasr31:9090/admin/secure/logon.do
Home | Save | Preferences | Logout | Help
Apply
OK
Reset
Cancel
WebSphere Status
February 11, 2003 8:50:22 AM EST

Figure 12-26 Create a reference

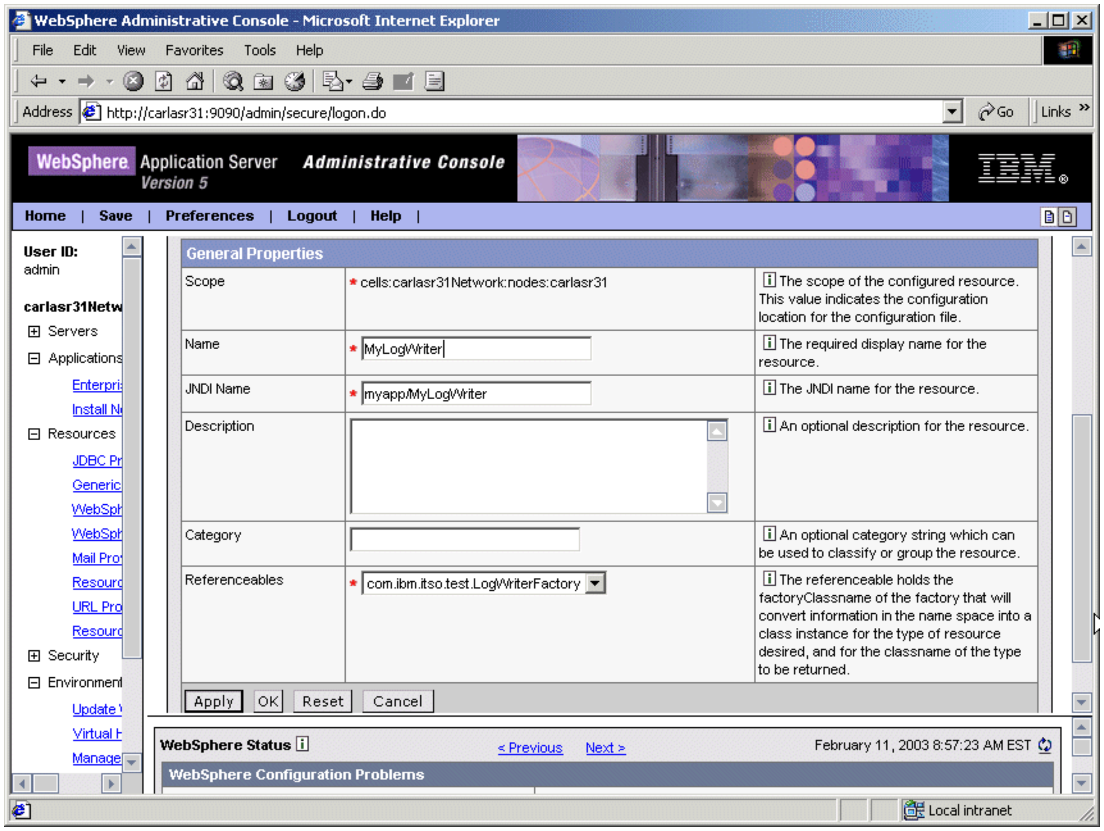
– **Factory Classname**

  Contains a javax.naming.ObjectFactory implementation class name.

– **Classname**

  The Java type that a referenceable provides access to, for binding validation and to create the reference data type string.

7. Click **OK**.
8. Select the resource environment provider (in the top navigation path) and click **Resource Env Entries** in the Additional Properties table.
9. Click **New**.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help |
User ID: admin
carlasr31Netw
Servers
Applications
Enterpri
Install N
Resources
JDBC Pr
Generic
WebSph
WebSph
Mail Pro
Resourc
URL Pro
Resourc
Security
Environment
Update
Virtual H
Manage
Apply
OK
Reset
Cancel
WebSphere Status
< Previous
Next >
February 11, 2003 8:57:23 AM EST
WebSphere Configuration Problems
Local intranet

General Properties

Scope	* cells:carlasr31Network:nodes:carlasr31	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* MyLogWriter	The required display name for the resource.
JNDI Name	* myapp/MyLogWriter	The JNDI name for the resource.
Description		An optional description for the resource.
Category		An optional category string which can be used to classify or group the resource.
Referenceables	* com.ibm.itso.test.LogWriterFactory	The referenceable holds the factoryClassname of the factory that will convert information in the name space into a class instance for the type of resource desired, and for the classname of the type to be returned.

Figure 12-27 Creating a resource environment entry

– **Name**

  (Required) A display name for the resource.

– **JNDI name**

  (Required) The JNDI name for the resource, including any naming subcontexts.

  This name is used as the link between the platform's binding information for resources defined by a module's deployment descriptor and actual resources bound into JNDI by the platform.

– **Description**

  A text description for the resource.

– **Category**

  A category string that can be used to classify or group the resource.

   – **Referenceable**

     The referenceable holds the factoryClassname of the factory that will convert information in the name space into a class instance for the type of resource desired, and for the classname of the type to be returned. Data type: Dropdown menu.

10. Click **OK**.
11. Save your configuration.

12.9.3 More information

These documents and Web sites are also relevant as further information sources:

- WebSphere InfoCenter. Search on Resource environment provider.

  `http://www.ibm.com/software/webservers/appserv/infocenter.html`

- J2EE 1.3 documentation.

  `http://java.sun.com/j2ee/1.3/docs/`

Part 4

Deploying applications

This part takes you through the process of preparing and deploying an application to a WebSphere Application Server environment. It uses a sample application to illustrate how to use the Application Server Toolkit to prepare the application, then the WebSphere administrative console to deploy the application.

13 WebSphere specific packaging options

This chapter provides the information needed to package a J2EE application for deployment into the WebSphere Application Server using the Assembly Toolkit feature of the Application Server Toolkit.

With the Assembly Toolkit, you can create and modify J2EE applications and modules, edit deployment descriptors, and map databases. This chapter focuses on modifying the IBM extensions to the standard J2EE deployment descriptors, such as EJB caching options or access intents. The assumption is that the development team has provided the WebSphere administrator with an application that is ready to deploy. The administrator simply needs to modify the deployment descriptors to match the runtime environment.

The Application Server Toolkit is basically a subset of the functions you would find in WebSphere Studio Application Developer. If you are familiar with the WebSphere Studio tools, you should have no problem using the toolkit. For a summary of the functions provided in the toolkit and information on how to use it, see Appendix B, “Application Server Toolkit” on page 879. For information on building and testing applications using WebSphere Studio, we recommend that you refer to *WebSphere Studio Application Developer Version 5 Programming Guide*, SG24-6957.

13.1 Webbank application overview

To illustrate the use of the Assembly Toolkit to prepare an application for deployment, we will assume the development team has provided the production team with an enterprise application (EAR file) for the sample Webbank application, ready to be packaged and deployed. The development team has used a development tool such as WebSphere Studio to develop and test the application.

The Webbank application is a simple application intended to illustrate concepts. It uses servlets, JSP pages, and enterprise beans and comes with stand-alone clients. The application allows you to transfer money between bank accounts and to view bank account balances. It is not a real-world application. The Webbank application is described in detail in Appendix D, “Webbank application overview” on page 917. We recommend that you read this appendix to understand the application design and implementation.

It is important for the rest of this chapter that you understand the structure of the Webbank application from a packaging point of view. The Webbank application is composed of five modules:

- A Web module containing the presentation code for the application
- Two EJB modules, one containing the session beans (business logic) and a second containing the entity beans (data access logic)
- Two client modules containing Java clients, one for transferring money between accounts and a second for viewing account balances

If you list the contents of the WebbankV51.ear file with the **`jar -tvf`** command, you will see something like this:

Example 13-1 Examining the WebbankV51.ear file

```
jar.exe -tvf WebbankV51.ear
347 Sun Feb 08 16:40:56 CET 2004 META-INF/ibm-application-ext.xmi
159857 Sun Feb 22 21:30:04 CET 2004 webbankEntityEJBs.jar
12978 Sun Feb 22 21:30:04 CET 2004 webbankConsultationClient.jar
14948 Sun Feb 22 21:30:04 CET 2004 webbankTransferClient.jar
1122601 Sun Feb 22 21:30:04 CET 2004 webbankWeb.war
103175 Sun Feb 22 21:30:06 CET 2004 webbankSessionEJBs.jar
12083 Sun Feb 22 21:30:06 CET 2004 webbankCommon.jar
25 Sun Feb 22 21:30:06 CET 2004 META-INF/MANIFEST.MF
780 Sun Feb 22 21:30:06 CET 2004 META-INF/application.xml
```

You can see all the modules we mentioned plus a webbankCommon.jar file, which contains common code used by all modules. XML files located under the META-INF directory in each module contain the standard J2EE deployment descriptors. XMI files located in the same folder contain IBM-specific extension descriptors (-ext.xmi files), or bindings (-bnd.xmi files).

13.2 Importing the Webbank application

To be able to work on the application packaging, you must first import it into the Application Server Toolkit.

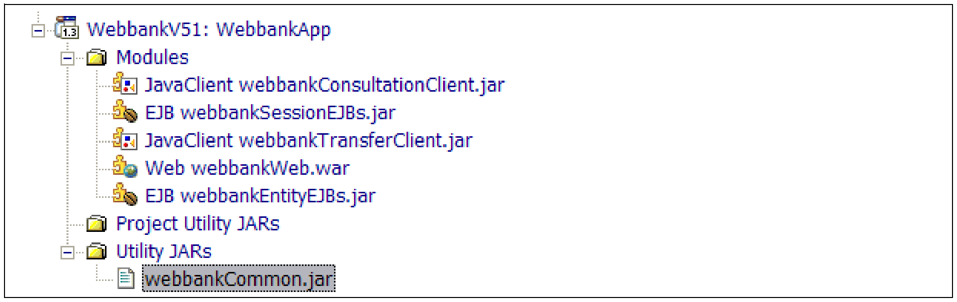
1. Start the toolkit.

> **Tip:** As you import the code, the standard J2EE deployment descriptors will be analyzed by validators in the toolkit. Set the auto-validation of the code to `false` before importing the code in a new workspace. This makes the import run faster. This can be done in **Window -> Preferences**, by deselecting the **Run Validation Automatically...** option in the Validation section.

2. Select **File -> Import**.
3. Select **EAR file** from the list and click **Next**.
4. Click **Browse...**, select the **WebbankV51.ear** file, and click **Open**.
5. Click **Finish**. This will import the file using the default options.

   Note that if you click **Next** instead of **Finish**, you can see options used for the import. Leave all options unchanged. In particular, do not select the **Optimized** option for Project Import, as this would prevent you from being able to edit the deployment descriptors.

Once the code has been imported, switch to the J2EE perspective (**Window -> Open Perspective -> J2EE Perspective)**. The J2EE Hierarchy view (Figure 13-1) shows the EAR file contents organized by module.



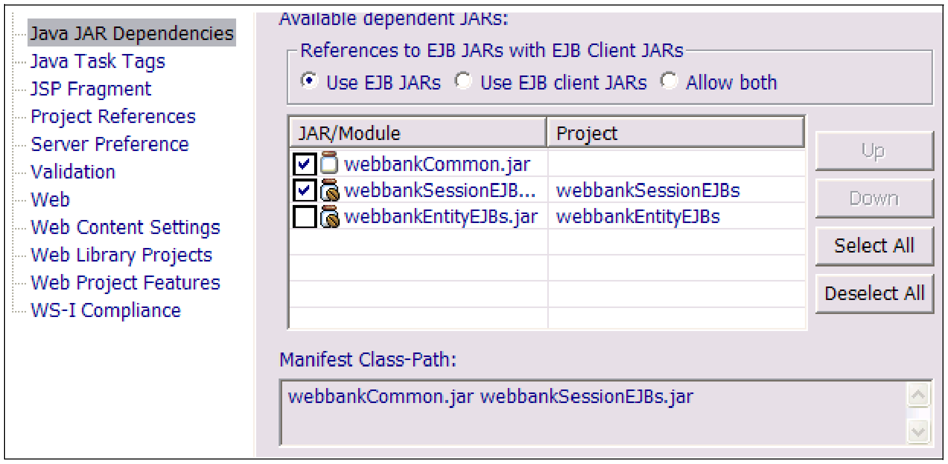
```
WebbankV51: WebbankApp
  Modules
    JavaClient webbankConsultationClient.jar
    EJB webbankSessionEJBs.jar
    JavaClient webbankTransferClient.jar
    Web webbankWeb.war
    EJB webbankEntityEJBs.jar
  Project Utility JARs
  Utility JARs
    webbankCommon.jar
```

Figure 13-1 Webbank Modules list

As you can see, the webbankCommon.jar file is listed under the Utility JARs category. This means that it is a special JAR file that can be used by the modules in the EAR (in other words, other modules use the classes from that JAR file). When a JAR file is referenced as a utility JAR, you can easily define that a module depends on that JAR. By "depends," we mean that it needs the classes contained in that JAR at runtime.

Let's take an example:

1. Navigate in the J2EE Hierarchy view, and select the **webbankWeb** module.
2. Open its properties page by right-clicking the Web module name and selecting **Properties...**.
3. Select the **Java JAR Dependencies** entry. You will see something similar to Figure 13-2. As you can see, the Web module depends on the webbankCommon.jar (for the common code) and the webbankSessionEJBs.jar (for the business logic code). How is this deduced in the application packaging? In reality, this page is an editor for the MANIFEST.MF file of the webbankWeb.war file. It tells the classloader of the Web module that those JAR files should be put on the Web module classpath. This is explained in more detail in 14.7.2, "Step 2: Sharing the dependency JAR among multiple modules" on page 699.



Available dependent JARs:

References to EJB JARs with EJB Client JARs

[✓ Use EJB JARs] Use EJB client JARs Allow both

JAR/Module	Project
✓ webbankCommon.jar	
✓ webbankSessionEJB...	webbankSessionEJBs
webbankEntityEJBs.jar	webbankEntityEJBs

Manifest Class-Path:

webbankCommon.jar webbankSessionEJBs.jar

Figure 13-2 Editing JAR dependencies

13.3 Working with deployment descriptors

Multiple IBM extensions are available to influence runtime behavior. These extensions are stored in descriptor files that basically supplement the EAR, EJB or Web deployment descriptors.

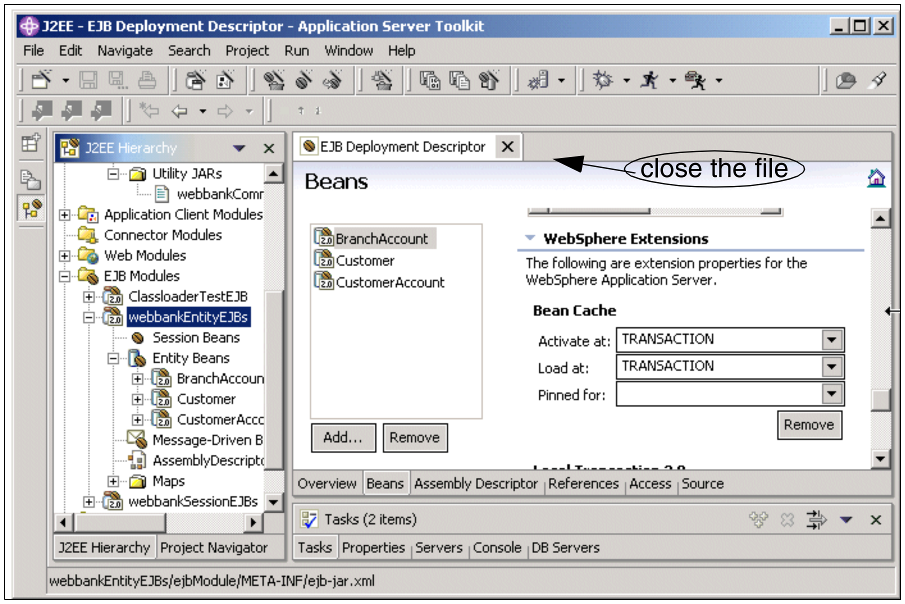
The toolkit makes it easy to work with deployment descriptors. When you open an EJB, EAR, or Web deployment descriptor, the Deployment Descriptor Editor combines the contents of that deployment descriptor and any relevant descriptors into one single GUI interface.

For example, if you open the EJB deployment descriptor, you will see settings that are stored across multiple deployment descriptors for the EJB module, including:

- The EJB deployment descriptor, ejb-jar.xml
- The extensions deployment descriptor, ibm-ejb-jar-ext.xmi
- The bindings file, ibm-ejb-jar-bnd.xmi files
- The access intent settings, ibm-ejb-access-bean.xmi

To work with a deployment descriptors:

1. Open the J2EE perspective.
2. In the J2EE Hierarchy view, double-click the module for which you want the descriptors. For example, if you double-click the EJB module name, this will open the EJB deployment descriptors in the editor area.



WebSphere Extensions

The following are extension properties for the WebSphere Application Server.

Bean Cache

Activate at: TRANSACTION

Load at: TRANSACTION

Pinned for:

Remove

Figure 13-3 Setting the activate and load settings for entity beans

3. You will see the content broken down into different tabs. Switch to each tab to view or modify the contents.
4. Close and save the deployment descriptor (click the **X** to the right of the file name at the top of the window).

Note that you can view the deployment descriptor source using the Source tab. However, keep in mind that the editor is showing you the settings stored in more than one file. The source you will see is the source of the deployment descriptor itself, for example, ejb-jar.xml or web.xml. Extensions and bindings are not set in this file. If you want to view the source results of updates you have made using the editor, you will need to open each file individually.

To find the descriptor files, go to the Project Navigator view and expand the module. You will find the EJB deployment descriptors in the ejbModule/META-INF directory of the EJB module. Web deployment descriptors are found in the WebContent/WEB-INF directory of the Web module. The application deployment descriptors are in the META-INF directory of the EAR module.

13.4 IBM EJB extensions: EJB caching options

This section will discuss the caching options for entity and stateful session beans.

13.4.1 EJB container caching option for entity beans

The Enterprise JavaBeans specification defines three EJB caching options: options A, B, or C. Those options define how the EJB container handles entity bean instances between transactions. EJB caching options are set at the bean level, and are part of the IBM extensions deployment descriptor.

Caching option A

With caching option A, you assume that the entity bean has *exclusive* access to the underlying persistent store. In other words, between transactions, no one can modify the data. This includes a batch program updating the data, a Java application updating the data, or even the same entity bean running in a *different* container. This implies option A cannot be used in a clustered environment (WLM). Note that it is your responsibility to ensure no other application will modify the data, as the EJB container has no way to control write access to the underlying database from other servers.

When caching option A is used, the entity bean instance is kept in a memory cache across transactions. At transaction commit, the entity bean attributes are synchronized with the underlying persistent store, and the bean instance remains cached in memory.

If you were tracing the calls made by the container, you would see something similar to Example 13-2. The first time the entity bean is used, its runtime context is set (step 1), a bean is taken from the entity beans instance pool (step 2), the bean instance attributes are synchronized with the underlying data store (step 3), the method setBalance is invoked on the bean (step 4), and finally the bean attributes are saved back to the database (step 5). The bean is not returned to the pool. On subsequent calls, the setBalance method is invoked directly on the cached bean instance, and the bean attributes are synchronized with the underlying persistent data store.

Example 13-2 Entity beans call trace with option A caching

```
Transaction 1 (Begin)
Step 1: 1c9585f1 BranchAccount E called setEntityContext() method
Step 2: 1c9585f1 BranchAccount E called ejbActivate() method
Step 3: 1c9585f1 BranchAccount E called ejbLoad() method
Step 4: 1c9585f1 BranchAccount E called setBalance() method
Step 5: 1c9585f1 BranchAccount E called ejbStore() method
Transaction 1 (Commit)
Transaction 2 (Begin)
Step 1: 284485f1 BranchAccount E called setBalance() method
Step 2: 284485f1 BranchAccount E called ejbStore() method
Transaction 2 (Commit)
```

Using caching option A can provide some performance enhancements at the expense of higher memory usage. You should only use it if you do not intend to use WebSphere clustering capabilities and you mostly access data in read mode.

Caching option B

With caching option B, you assume that you have *shared* access to the underlying database, which means the data could be changed by another application between transactions. When option B is used, the bean instance attributes are always synchronized with the underlying back-end data store at the beginning of every transaction. Similar to Option A, the bean is kept in the cache between transactions. Therefore, if you were tracing the different calls made in Option B, you would obtain the trace shown in Example 13-3.

Example 13-3 Entity beans call trace with option B caching

```
Transaction 1 (Begin)
Step 1: 1c9585f1 BranchAccount E called setEntityContext() method
Step 2: 1c9585f1 BranchAccount E called ejbActivate() method
Step 3: 1c9585f1 BranchAccount E called ejbLoad() method
Step 4: 1c9585f1 BranchAccount E called setBalance() method
Step 5: 1c9585f1 BranchAccount E called ejbStore() method
Transaction 1 (Commit)

Transaction 2 (Begin)
Step 1: 284485f1 BranchAccount E called ejbLoad() method
Step 2: 284485f1 BranchAccount E called setBalance() method
Step 3: 284485f1 BranchAccount E called ejbStore() method
Transaction 2(Commit)
```

Caching option B can be safely used in a clustered environment, or when you are not sure if you have exclusive access to data. You are assured that you always work with the last committed data. Option B memory usage is the same as for option A. The performance of both options may slightly differ depending on the nature of your application.

Caching option C

Similar to option B, caching option C assumes *shared* access to the database. Unlike option B or A, the bean instance is returned to the entity beans pool at the end of the transaction. A new bean instance is used at the beginning of every transaction. Each transaction results in the sequence of calls shown in Example 13-4.

Example 13-4 Entity beans call trace with option C caching

```
Transaction (Begin)
Step 1: 1c9585f1 BranchAccount E called setEntityContext() method
Step 2: 1c9585f1 BranchAccount E called ejbActivate() method
Step 3: 1c9585f1 BranchAccount E called ejbLoad() method
Step 4: 1c9585f1 BranchAccount E called setBalance() method
Step 5: 1c9585f1 BranchAccount E called ejbStore() method
Step 6: 1c9585f1 BranchAccount E called ejbPassivate() method
Step 7: 1c9585f1 BranchAccount E called unsetEntityContext() method
Transaction (Commit)
```

Caching option C has the best memory usage at the expense of a larger number of methods calls. This is the default behavior.

How to set the EJB caching option

You must combine the *Activate at* and *Load at* options to set the EJB caching option to A, B, or C. Use Table 13-1 to choose the right combination.

Table 13-1 Setting entity EJB caching properties

Option	**Activate at must be set to**	**Load at must be set to**
Option A	Once	Activation
Option B	Once	Transaction
Option C (default)	Transaction	Transaction

1. Open the EJB deployment descriptor.
1. Switch to the Beans tab.
2. Select the entity bean in the window to the left, then scroll down the options at right until you see the WebSphere extensions settings.

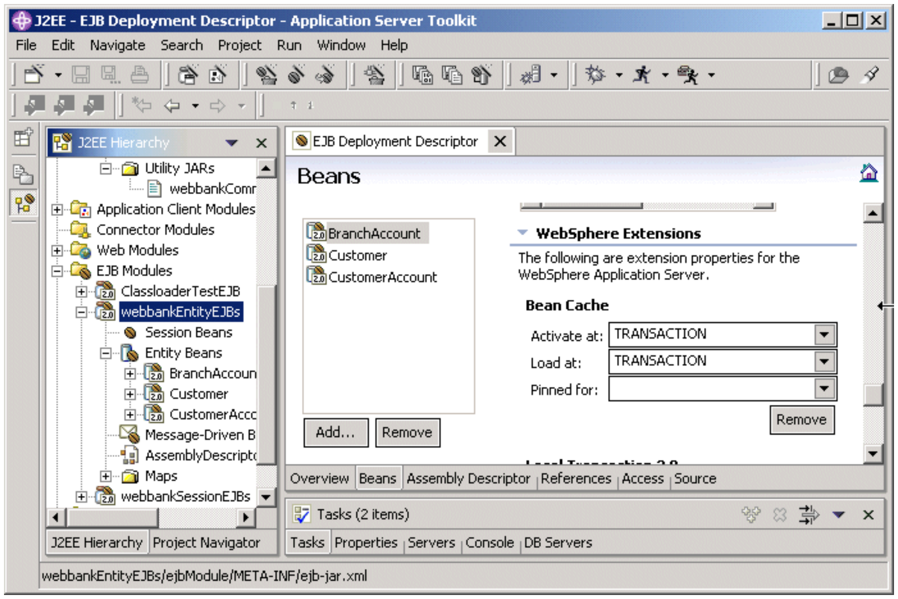

J2EE - EJB Deployment Descriptor - Application Server Toolkit
File Edit Navigate Search Project Run Window Help
J2EE Hierarchy
Utility JARs
webbankComr
Application Client Modules
Connector Modules
Web Modules
EJB Modules
ClassloaderTestEJB
webbankEntityEJBs
Session Beans
Entity Beans
BranchAccoun
Customer
CustomerAcc
Message-Driven B
AssemblyDescripto
Maps
webbankSessionEJBs
J2EE Hierarchy Project Navigator
EJB Deployment Descriptor
Beans
BranchAccount
Customer
CustomerAccount
Add... Remove
WebSphere Extensions
The following are extension properties for the WebSphere Application Server.
Bean Cache
Activate at: TRANSACTION
Load at: TRANSACTION
Pinned for:
Remove
Overview Beans Assembly Descriptor References Access Source
Tasks (2 items)
Tasks Properties Servers Console DB Servers
webbankEntityEJBs/ejbModule/META-INF/ejb-jar.xml

Figure 13-4 Setting the activate and load settings for entity beans

3. Select the activate and load options according to Table 13-1 on page 625.
4. Close and save the deployment descriptor.

The settings are saved in the ibm-ejb-jar-ext.xmi file. They correspond to the following entry (one line per entity bean you have set this option for):

```
<beanCache xmi:id="BeanCache_1" activateAt="ONCE" loadAt="ACTIVATION"/>
```

13.4.2 EJB container caching option for stateful session beans

Similarly to entity beans, you can specify which caching strategy should be used for stateful session beans. This caching option specifies the point at which an enterprise bean is activated and placed in the cache. Removal from the cache and passivation are also governed by this setting. Valid values are:

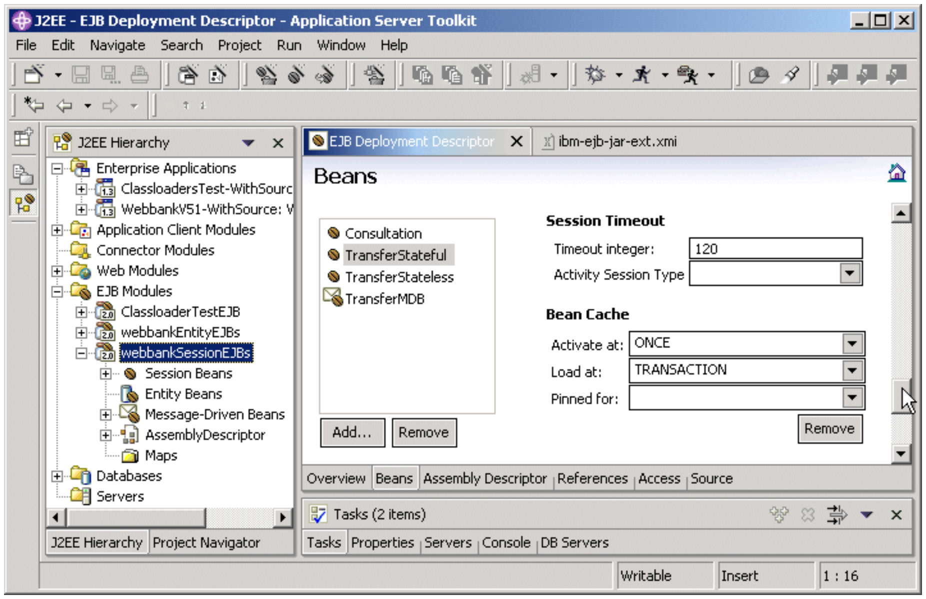
- Once (default)

  Indicates that the bean is activated when it is first accessed in the server process. It is passivated (and removed from the cache) at the discretion of the container, for example, when the cache becomes full.

- Transaction

  Indicates that the bean is activated at the start of a transaction and passivated (and removed from the cache) at the end of the transaction.

You can set this caching option by opening the EJB deployment descriptor for the EJB module. The *Activate at* setting is found on the Bean tab (Table 13-5). Select the bean and scroll down to the Bean Cache category.



Session Timeout

Timeout integer: 120

Activity Session Type:

Bean Cache

Activate at: ONCE

Load at: TRANSACTION

Pinned for:

Figure 13-5   Activate settings for stateful session beans

13.4.3 Stateful EJB timeout option

Additionally, you can specify a timeout value for stateful session beans. A bean can time out in the METHOD_READY or in the PASSIVATED state. If you try to access a bean that has timed out, you will see an exception similar to:

```
com.ibm.ejs.container.SessionBeanTimeoutException: Stateful bean
StatefulBeanO(BeanId(Webbank#webbankEJBs.jar#Transfer, ebf64d846a), state =
METHOD_READY) timed out.
```

Session beans that have timed out can be removed by the container, for example if it needs to free memory. However, a well-written application should not rely on beans to time out to free memory. Instead, it is important that the developer explicitly calls remove() on a bean when this stateful bean is not needed anymore.

The default timeout is 600 seconds. You can set the timeout integer value as a parameter of a stateful session bean by opening the EJB deployment descriptor and selecting the **Beans** tab (Table 13-5 on page 627).

By specifying a value of `0`, you set the bean to never expire. Setting this timeout inserts the following property in the ejbExtensions tag of the IBM bindings file:

```
<ejbExtensions xmi:type="ejbext:SessionExtension"
   xmi:id="Session_1_Ext" timeout="120">
```

Note: If a bean times out in the METHOD_READY state and is consequently removed by the container, the ejbRemove() method will be called on the bean instance. If a bean times out in the passivated state, ejbRemove() is not called (as per the EJB specification).

13.5 IBM EJB extensions: EJB access intents

Access intents are used by the persistence manager to optimize the access to relational data. Version 5 brings new flexibility in managing access to data, and drives things such as transaction isolation levels or database locks. Different levels of service are available depending on whether you use the EJB 1.1 or EJB 2.0 container.

For EJBs at the 1.1 level, you can set a transaction isolation level at the method level, as well as mark methods as read-only, or tag methods as being used to drive a database update. You can also define which type of concurrency control you want to use (pessimistic or optimistic mode).

For EJBs at the 2.0 level, you must use access intents. Access intent policies are specifically designed to supplant the use of isolation level and read-only, method-level modifiers found in the extended deployment descriptor for EJB Version 1.1 enterprise beans. You cannot specify isolation level and read-only modifiers for EJB Version 2.0 enterprise beans. The WebSphere persistence manager uses access intent hints to make decisions about isolation level, cursor management, and more.

13.5.1 Transaction isolation levels overview

Transaction isolation levels provides a trade-off between accuracy of reads versus concurrent readers. The levels can best be described by the types of read anomalies they permit and forbid. Consider the read anomalies that can occur with two concurrent transactions, T1 and T2:

- Dirty read: T1 reads data that has been modified by T2, before T2 commits.

- Non-repeatable read: This is caused by fine-grained locks.
  - T1 reads a record and drops its lock.
  - T2 updates.
  - T1 re-reads different data.
- Phantom read: A non-repeatable read involving a range of data and inserts or deletes on the range.
  - T1 reads a set of records that match some criterion.
  - T2 inserts a record that matches the criterion.
  - T1 continues processing the set, which now includes records that were not part of the original matching set.

There are four possible settings for the transaction isolation level:

- Repeatable read (`TRANSACTION_REPEATABLE_READ`)

  Permits phantom reads and forbids both dirty and unrepeatable reads.
- Read committed (`TRANSACTION_READ_COMMITTED`)

  Permits non-repeatable and phantom reads and forbids dirty reads.
- Read uncommitted (`TRANSACTION_READ_UNCOMMITTED`)

  Permits all the read anomalies including dirty reads, non-repeatable reads, and phantom reads.
- Serializable (`TRANSACTION_SERIALIZABLE`)

  Forbids all the read anomalies.

The container applies the isolation level as follows:

- For entity beans with Container Managed Persistence (CMP), the container generates code that assures the desired level of isolation for each database access.
- For session beans and Bean Managed Persistence (BMP) entity beans, the container sets the isolation level at the start of each transaction, for each database connection.

The transaction isolation level is tied to a database connection. The connection uses the isolation level specified in the first bean that uses the connection. The container throws an IsolationLevelChangeException whenever the connection is used by another bean method that has a different isolation level.

Not all databases support all JDBC isolation levels. Moreover, JDBC definitions for isolation levels may not match the database definition of isolation levels. As an example, DB2 definitions for isolation levels follow the naming conventions used in Jim Gray's classic book on transaction processing, *Transaction*

Processing: Concepts and Techniques. Table 13-2 shows a mapping between EJB and DB2 isolation levels.

Table 13-2 Mapping JDBC isolation levels to DB2 isolation levels

JDBC isolation level	DB2 isolation level
TRANSACTION_SERIALIZABLE	Repeatable Read
TRANSACTION_REPEATABLE_READ	Read Stability
TRANSACTION_READ_COMMITTED	Cursor Stability
TRANSACTION_READ_UNCOMMITTED	Uncommitted Read

To learn more, you should refer to the documentation provided by the database product.

13.5.2 Concurrency control

Concurrency control is the management of contention for data resources. A concurrency control scheme is considered pessimistic when it locks a given resource early in the data-access transaction and does not release it until the transaction is closed. A concurrency control scheme is considered optimistic when locks are acquired and released over a very short period of time at the end of a transaction.

The objective of optimistic concurrency is to minimize the time over which a given resource would be unavailable for use by other transactions. This is especially important with long-running transactions, which under a pessimistic scheme would lock up a resource for unacceptably long periods of time.

WebSphere uses *an overqualified update scheme* to test whether the underlying data source has been updated by another transaction since the beginning of the current transaction. With this scheme, the columns marked for update and their original values are added explicitly through a WHERE clause in the UPDATE statement so that the statement fails if the underlying column values have been changed. As a result, this scheme can provide column-level concurrency control; pessimistic schemes can control concurrency at the row level only.

Optimistic schemes typically perform this type of test only at the end of a transaction. If the underlying columns have not been updated since the beginning of the transaction, pending updates to container-managed persistence fields are committed and the locks are released. If locks cannot be acquired or if some other transaction has updated the columns since the beginning of the current transaction, the transaction is rolled back: all work performed within the transaction is lost.

Pessimistic and optimistic concurrency schemes require different transaction isolation levels. Enterprise beans that participate in the same transaction and require different concurrency control schemes cannot operate on the same underlying data connection unless the connection is able to change its isolation level on an individual-query basis. Some, but not all, JDBC drivers can do this. For those JDBC drivers that cannot, mixing concurrency controls requires the use of multiple connections within a transaction.

Whether or not to use optimistic concurrency depends on the type of transaction. Transactions with a high penalty for failure might be better managed with a pessimistic scheme. For low-penalty transactions, it is often worth the risk of failure to gain efficiency through the use of an optimistic scheme.

13.5.3 Setting isolation levels (EJB 1.1 only)

Isolation levels can be configured by opening the EJB deployment descriptor using the Deployment Descriptor Editor and selecting the **Access** page (Figure 13-6 on page 632).

These can be set at the bean level (`* - All methods` option), at the home methods level (`* - Home Methods` option), at the remote methods level (`* - Remote methods` option) or at the individual method level option.

13.5.4 Marking methods as read-only (EJB 1.1 only)

When you invoke an entity bean business method, the EJB container assumes that changes have been made to the bean attributes and systematically synchronizes the bean attributes with the persistent data store. When a method is marked as read-only, the container skips the STORE operation at the end of a transaction (regardless of the EJB caching option you are using, A, B, or C).

Note: For EJBs with container-managed persistence at the 2.0 level, this is taken into account in the concrete implementation of the bean. A bean will be marked as “dirty” whenever a setMethod is used. Only dirty beans will be synchronized with the underlying data store. Note that for EJBs with bean-managed persistence, you will have to handle this in the code, for example by using a dirty flag.

You should mark all the getXXX() methods of your entity beans as being read-only. This avoids possible read-to-write lock promotions and can significantly improve performance and concurrency. Note that it is your responsibility to ensure that methods you mark as read-only actually *are* read-only methods. The EJB container has no way of knowing this. Moreover, you should use this flag *only* for your business methods.

Access intent and isolation levels can be configured by opening the EJB deployment descriptor using the Deployment Descriptor Editor and selecting the **Access** page (Figure 13-6).

To set the read-only flags on methods:

1. Click the **Add** button and select **Read**.
2. Click **Next** and choose the bean(s) containing the method(s).
3. Click **Next** and choose the method(s) to apply this setting to.
4. Click **Finish**.

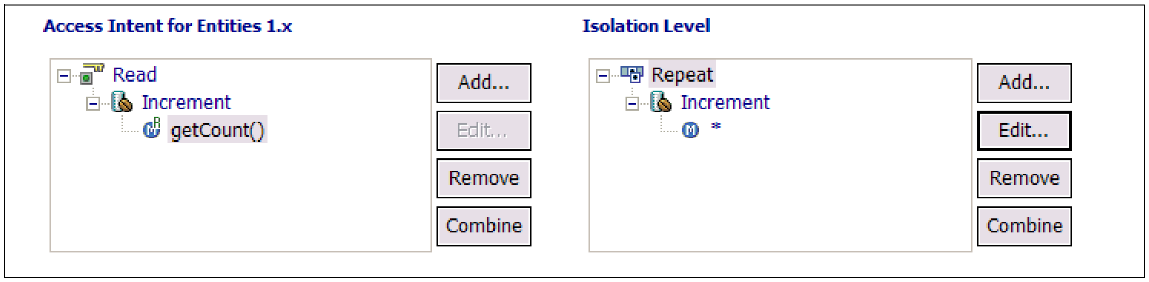


Figure 13-6 Setting access intents on EJB 1.1 beans

13.5.5 Marking methods with forUpdate flag (EJB 1.1 only)

This flag can be used to force SELECT calls to be done with a FOR UPDATE option. This will set a stronger lock at the database level; this lock will allow updates to data. If you do a simple SELECT, then a lock escalation has to be done by the database if you try to do an UPDATE call in the same transaction. This will typically happen if you write something like:

```
BranchAccountLocal branchAcct = findBranchAccount(branchAcctKey);
branchAcct.debit(amountCent);
```

The find call will typically trigger a SELECT query, taking a read lock on the database. The debit method, however, needs to update data, and therefore needs a stronger lock to preserve data integrity. If another transaction has a lock on the same row or table, the current transaction will have to wait. If the other transaction does a similar thing (both want to escalate their locks), you are in a deadlock situation.

By using the forUpdate flag, you immediately take a write lock instead of a read lock by executing a SELECT FOR UPDATE. No lock escalation needs to occur, which prevents potential deadlocks.

Access intent and isolation levels can be configured by opening the EJB deployment descriptor using the Deployment Descriptor Editor and selecting the **Access** page (Figure 13-6 on page 632).

To set the update flags on methods:

1. Click the **Add** button and select **Update**.
2. Click **Next** and choose the bean(s) containing the method(s).
3. Click **Next** and choose the method(s) to apply this setting to.
4. Click **Finish**.

13.5.6 Using EJB 2.0 access intents

Access intents policies are new to WebSphere Application Server V5. They let you define in a very flexible and powerful way how relational data will be accessed, should you be using BMP or CMP entity beans. Access intents cover the problems described above, such as the transaction isolation level or the find forUpdate flag.

Important: Although you can set access intents on a BMP, you are actually responsible for reading the access intent metadata from your code and applying the corresponding change to the isolation levels yourself (by calling `connection.setTransactionLevel()`). The EJB container has no control over your persistence strategy, and therefore cannot perform the same tasks as it can for CMPs. Access intent data is valid in the WebSphere naming service for the time of the transaction. See the InfoCenter for more coding examples.

Access intents policies

Seven access intent policies are available. They cover a wide variety of ways to access data. They are summarized in Table 13-3.

Table 13-3 Access intent policies

Access Intent Policy	Concurrency control	For Update used	Transaction isolation level	Notes
wsPessimisticRead	pessimistic	No	read committed	Read locks are held for the duration of the transaction. Updates are not permitted; the generated SELECT query does not include FOR UPDATE

Access Intent Policy	Concurrency control	For Update used	Transaction isolation level	Notes
wsPessimisticUpdate	pessimistic	Yes	For Oracle, read committed. Otherwise, repeatable read	The generated SELECT FOR UPDATE query grabs locks at the beginning of the transaction
wsPessimisticUpdate-Exclusive	pessimistic	Yes	serializable	SELECT FOR UPDATE is generated; locks are held for the duration of the transaction
wsPessimisticUpdate-NoCollision	pessimistic	No	read committed	The generated SELECT query does not include FOR UPDATE. *No locks are held, but updates are permitted.*
wsPessimisticUpdate-WeakestLockAtLoad (DEFAULT VALUE)	pessimistic	No (Oracle, yes)	For Oracle, read committed, otherwise repeatable read	For Oracle, this is the same as wsPessimisticUpdate. Otherwise, the generated SELECT query does not include FOR UPDATE; locks are escalated by the persistent store at storage time if updates were made.
wsOptimisticRead	optimistic	No	read committed	
wsOptimisticUpdate	optimistic	No	read committed	Generated overqualified-update query forces failure if CMP column values have changed since the beginning of the transaction

There are two critical questions you should ask yourself when using access intents:

- *At which point in my transaction do I access data?* This is critical in selecting which method you must set the access intent on.
- *How do I want to access data?* This is critical to selecting the best access intent to apply on the method.

Choosing where to apply the access intent

This is critical since it is at this point that the WebSphere persistence manager will decide which access intent to use. To illustrate this point, let’s use the following example. The Consultation session bean obtains the CustomerAccount balance using the following getBranchAccountBalance:

```
int getCustomerAccountBalance () {
...
custAcct = (CustomerAccountLocal)
custAcctHome.findByPrimaryKey(custAcctKey);
return custAcct.getBranchBalance();
}
```

Now, say you have applied AccessWriteIntent1 on the findByPrimarykey() method of the CustomerAccount bean, and AccessReadIntent2 on the getBranchBalance() method. Since the first access to the database in the transaction started by the call to getCustomerAccountBalance() is done by the findByPrimaryKey method, then AccessWriteIntent1 is used for *all calls* within the transaction. AccessReadIntent2 will be ignored by the persistence manager and therefore useless in this case.

This may be fine or not, depending on what you want to achieve. The critical point here is that you could use the findByPrimaryKey method in read or/and update transactions. If you use it in an update transaction, you probably want to execute it with, for example, a PessimisticUpdate intent. If you access data only for reading it, this would be overkill.

There are two main solutions you can adopt for this problem. The simplest one is to have two or more versions of your finder methods, specialized by access intent, such as findWhateverForRead and findWhateverForUpdate. If you look at the Webbank sample, you will see that all entity beans have a custom findByPrimaryKeyForRead() method and the standard findByPrimaryKey() method. You would set the access intent of this finder, say to wsOptimisticRead, and use it in all read transactions. However, you would use the standard findByPrimaryKey method whenever you know you are accessing an entity bean to update data. The default access intent (wsPessimisticUpdate-WeakestLockAtLoad) is probably fine in most cases for the findByPrimaryKey() method (as well as for other finder or non-finder methods).

Important: This solution is also well adapted to BMPs. By having a different findByPrimaryKey method for read and write transactions, you can easily set a different isolation level in the code for each of them. You can also define a different SELECT query (one with a FOR UPDATE clause and one without) and call them from those different methods.

Another solution is to run findByPrimaryKey (or another finder) in its own transaction. This is done by applying a RequiresNew transaction flag on it. Let's look at the sample above again:

1. The getCustomerAccountBalance method starts a new transaction.
2. findByPrimaryKey is called. The current transaction is paused, the findByPrimary method executes within its own transaction and therefore own access intent. The call to findByPrimaryKey() returns an un-hydrated instance (which means it has not been activated nor loaded).
3. The transaction initiated by the session bean resumes.
4. getBalance() is called on the instance returned by findByPrimaryKey. The instance is hydrated and the access intent specified for this method is used. Any other method calls within the transaction will execute with the same access intent.

Note: IBM WebSphere Application Server Enterprise provides an extension to access intents called Application Profiles, which handles the problem mentioned above in a powerful way. Application profiles let you externally specify a set of tasks (that is, a flow of calls in your code), and specify which access intent should be used for a specific task.

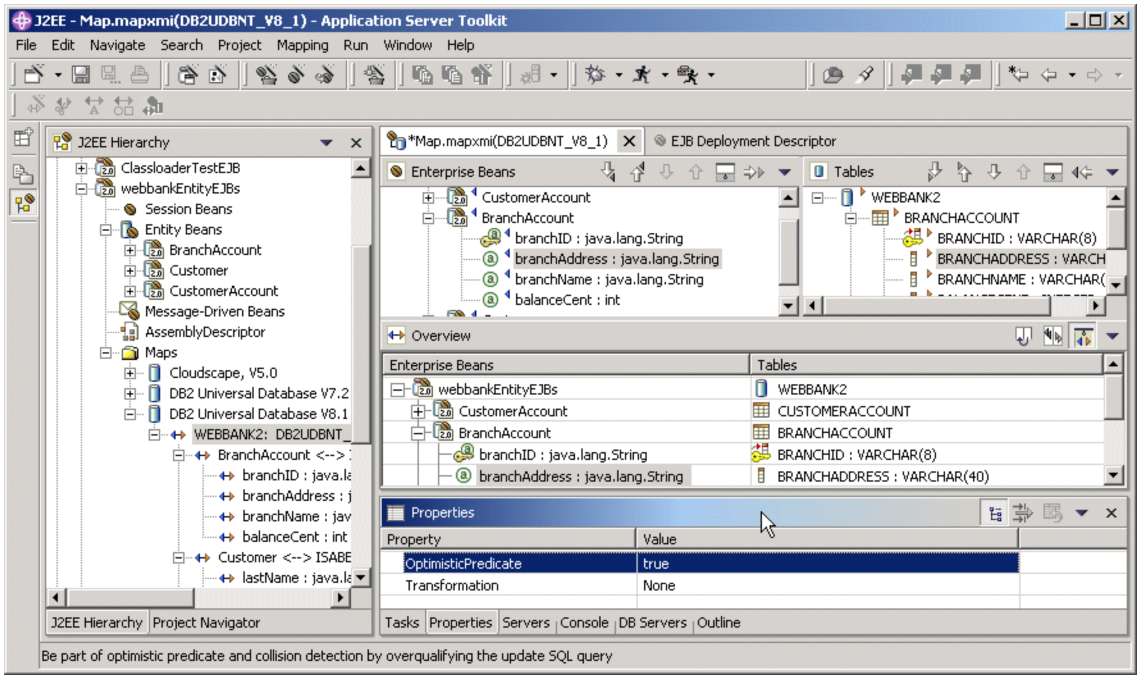
Choosing the right access intent

The main rule is: keep it simple. Start with the default setting (wsPessimisticUpdate-WeakestLockAtLoad), and work from there. Specifying access intents on all your business methods could lead to a configuration, debugging, and maintenance nightmare. Specify access intents on a selected number of methods. Also, choose access intents wisely.

- Access intents can be applied to your business methods, to the findByPrimaryKey() method, as well as the create and remove method. As much as possible, avoid other methods.
- Make sure that no method is configured with more than one access intent policy. Applications that are misconfigured in this way will not be runnable until the configuration errors are fixed.
- For entity beans that are backed by tables with nullable columns, use optimistic policies with caution. Nullable columns are automatically excluded from overqualified updates at deployment time. This means that at commit time, those columns will not be used in the update statement to check whether the data has changed or not. Therefore, concurrent changes to a nullable field might result in lost updates. Using the Assembly Toolkit, you can set a property on each enterprise bean attribute called *OptimisticPredicate*, as shown in Figure 13-7 on page 637. You can change this property by editing the data mappings of your EJBs. When this property is set, the

column, even if it is nullable, will be reflected in the overqualified update statement that is generated in the deployment code to support optimistic policies.

Tip: If you want to check which SQL code is executed for an optimistic update, check the storeUsingOCC method in the <beanname>FunctionSet generated class.



Property	Value
OptimisticPredicate	true
Transformation	None

Be part of optimistic predicate and collision detection by overqualifying the update SQL query

Figure 13-7 Optimistic predicate property

- If a bean is loaded for read intent and an update is attempted during that transaction, then the persistence manager will raise an UpdateCannotProceedWithIntegrity exception. In other words, if you call findWhateverForRead() and an update is attempted, it will fail.

Important: The behavior described above is true for all access intents *but* the wsPessimisticUpdate-NoCollision one! This access intent *will not* flag updates, even if no locks are held. Our recommendation is that you avoid this access intent in production.

13.5.7 Using read-ahead hints

Read-ahead schemes enable applications to minimize the number of database round trips by retrieving a working set of CMP beans for the transaction within one query. Read-ahead involves activating the requested CMP beans and caching the data for their related beans, which ensures that data is present for the beans that are most likely to be needed next by an application.

A read-ahead hint is a canonical representation of the related beans that are to be read. It is associated with a finder method for the requested bean type, which must be an EJB 2.x-compliant CMP entity bean. Currently, only findByPrimaryKey methods can have read-ahead hints. Only beans related to the requested beans by a container-managed relationship (CMR), either directly or indirectly through other beans, can be read ahead.

To set Read-ahead hints, follow these steps:

1. Open the EJB deployment descriptor using the Deployment Descriptor editor (double-click the EJB module).
2. Select the **Access** tab and scroll down to the WebSphere Extensions section.
3. Click the **Add** button to the right of the Access Intent for Entities 2.x (Method Level) field.

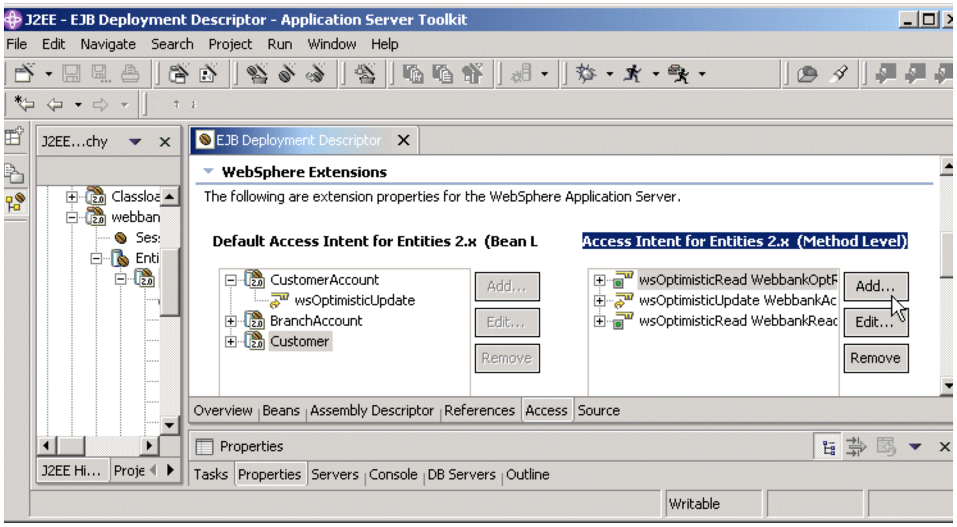
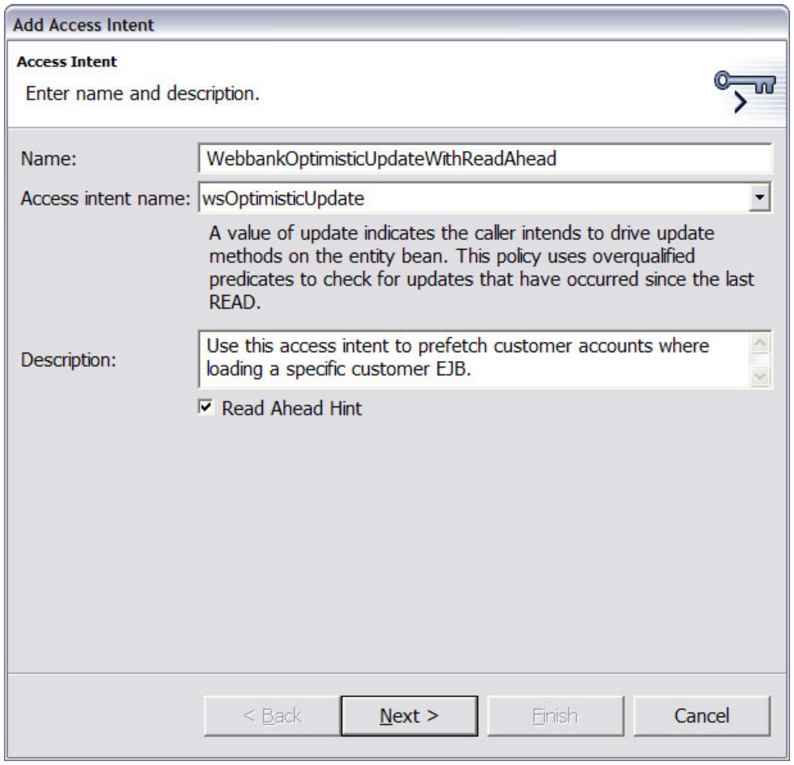

J2EE - EJB Deployment Descriptor - Application Server Toolkit
File Edit Navigate Search Project Run Window Help
J2EE...chy
EJB Deployment Descriptor
WebSphere Extensions
The following are extension properties for the WebSphere Application Server.
Default Access Intent for Entities 2.x (Bean L
Access Intent for Entities 2.x (Method Level)
CustomerAccount
wsOptimisticUpdate
BranchAccount
Customer
wsOptimisticRead WebbankOptF
wsOptimisticUpdate WebbankAc
wsOptimisticRead WebbankReac
Add...
Edit...
Remove
Overview Beans Assembly Descriptor References Access Source
Properties
Tasks Properties Servers Console DB Servers Outline
Writable

Figure 13-8 Adding a read ahead hint

In the wizard, the Read Ahead Hint check box is enabled only with access intent policies with optimistic concurrency. Read-ahead is limited to optimistic policies because locking persistent data store for all beans represented in the hint would be more likely to cause lock conflicts, and optimistic policies do not obtain locks until immediately before the database operation.



Name: WebbankOptimisticUpdateWithReadAhead

Access intent name: wsOptimisticUpdate

A value of update indicates the caller intends to drive update methods on the entity bean. This policy uses overqualified predicates to check for updates that have occurred since the last READ.

Description: Use this access intent to prefetch customer accounts where loading a specific customer EJB.

☑ Read Ahead Hint

Figure 13-9 Specifying read-ahead hint

4. Follow the panels in the wizard (select the beans and methods) and click **Finish**.

13.5.8 Tracing access intents behavior

You can obtain a very detailed trace of the EJB persistence manager behavior by specifying the following trace specification for an application server:

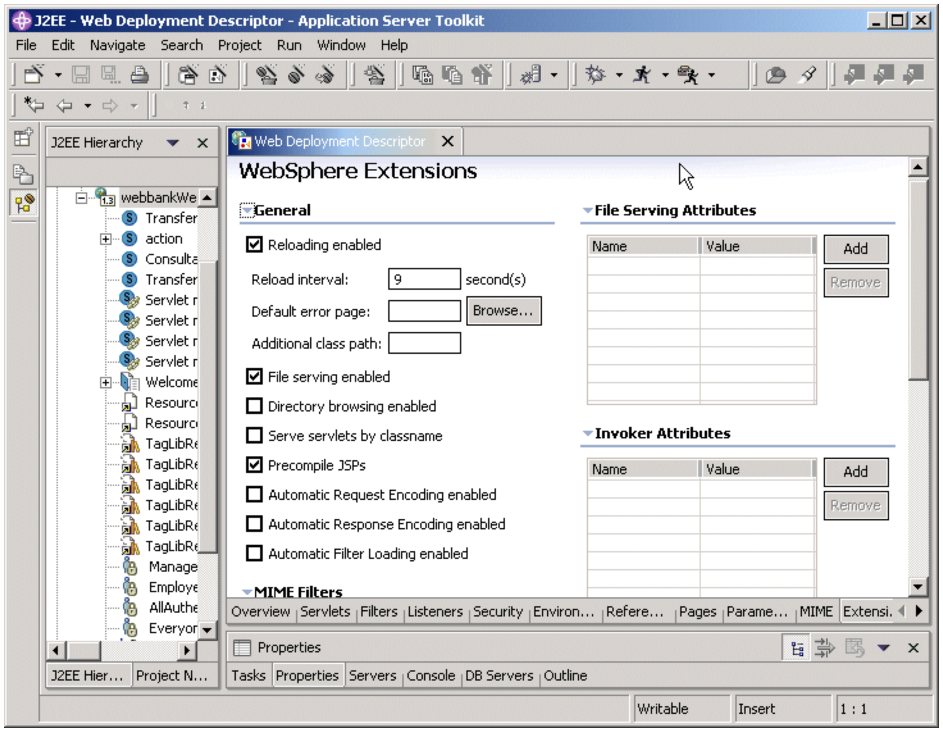
```
com.ibm.ejs.container.*=all=enabled:com.ibm.ejs.persistence.*=all=enabled:
com.ibm.ws.appprofile.*=all=enabled.
```

13.6 IBM EJB extensions: inheritance relationships

Support for entity beans inheritance, which is not part of the EJB 1.1 nor EJB 2.0 specifications, is also available via the toolkit. Support for enterprise entity beans relationships for EJB 1.1, although not standard, is also available using this tool. Refer to the toolkit documentation for more details.

13.7 IBM Web modules extensions

WebSphere Application Server V5 provides multiple extensions for Web modules. To work with these extensions, open the Web deployment descriptor by double-clicking the Web module in the J2EE Hierarchy view. To see the IBM Web module extensions, select the **Extensions** tab.


J2EE - Web Deployment Descriptor - Application Server Toolkit
File Edit Navigate Search Project Run Window Help
J2EE Hierarchy
Web Deployment Descriptor
File Serving Attributes
Invoker Attributes
Name Value
Add
Remove
MIME Filters
Overview Servlets Filters Listeners Security Environ... Refere... Pages Parame... MIME Extensi.
Properties
Tasks Properties Servers Console DB Servers Outline
Writable Insert 1 : 1

WebSphere Extensions

General

- [✓] Reloading enabled
- Reload interval: 9 second(s)
- Default error page:
- Additional class path:
- [✓] File serving enabled
- [ ] Directory browsing enabled
- [ ] Serve servlets by classname
- [✓] Precompile JSPs
- [ ] Automatic Request Encoding enabled
- [ ] Automatic Response Encoding enabled
- [ ] Automatic Filter Loading enabled

Figure 13-10 Web module extensions

13.7.1 File serving servlet

When dealing with static HTML pages, you can choose to have static pages served by WebSphere or have them served by the Web server itself.

If you want WebSphere to serve the static content of your application, you must enable the file servlet (also known as file serving servlet or file serving enabler). This servlet serves up any resource file packaged in the WAR file. The *File serving enabled* attribute is set to `true` by default, and should be set to `true` for the Webbank application.

For the case where HTML pages are served by the Web server, as opposed to being served by the WebSphere, you may experience better performance, since the Web server is serving the pages directly. Moreover, a Web server has much more customization options than the file servlet can offer. However, using the WebSphere file serving servlet has the advantage of keeping the static content organized in a single deployable unit with the rest of the application. Additionally, this allows you to protect the static pages using WebSphere security.

Refer to 14.13, “Separating static content from dynamic content” on page 710 to learn more about deploying the static content of an application to a Web server.

To enable this option, check the **File serving enabled** box. Attributes used by the file serving servlet can be specified in the *File Serving Attributes* section.

13.7.2 Web application auto reload

If you check the **Reloading enabled** option, the classpath of the Web application is monitored and all components (JAR or class files) are reloaded whenever a component update is detected (the Web module’s classloader is brought down and restarted). The reload interval is the interval between reloads of the Web application. It is set in seconds.

The auto reload feature plays a critical role in hot deployment/dynamic reload of your application.

Important: You must set the Reloading enabled option to `true` for JSP files to be reloaded when they are changed on the file system. Reloading a JSP does *not* trigger the reload of the Web module, since separate classloaders are used for servlets and JSP.

This option is set to `true` by default, with the reload interval set to `9` seconds. In production mode, you may consider making the reload interval much higher.

13.7.3 Serve servlets by class name

The invoker servlet can be used to invoke servlets by class name. Note there is a potential security risk with leaving this option set in production; it should be seen as more of a development-time feature, for quickly testing your servlets.

This option is off by default.

13.7.4 Default error page

This page will be invoked to handle errors if no error page has been defined, or if none of the defined error pages matches the current error.

13.7.5 Directory browsing

This boolean defines whether it is possible to browse the directory if no default page has been found.

This option is off by default.

13.7.6 JSP attributes

The following options can be set for the JSP compiler:

- keepgenerated

  If this boolean is set to `true`, the source code of the servlet created by compilation of a JSP page is kept on the file system. Otherwise, it is deleted as soon as the servlet code has been compiled (only the .class file is available).

- scratchdir

  This string represents the directory in which servlets code will be generated. If this string is not set, code is created under:

  ```
  <WAS_HOME>\temp\<hostname>\<application_server_name>\<applicationname>\<web
  modulename>.
  ```

13.7.7 Automatic HTTP request and response encoding

The Web container no longer automatically sets request and response encodings and response content types. The programmer is expected to set these values using the methods available in the Servlet 2.3 API. If you want the application server to attempt to automatically set these values, check the **Automatic Request Encoding enabled** option in order to have the request encoding value set. Similarly, you can check the **Automatic Response Encoding enabled** in order to have the response encoding and content type set.

The default value of the autoRequestEncoding and autoResponseEncoding extensions is false, which means that both the request and response character encoding is set to the Servlet 2.3 specification default of ISO-8859-1. Different character encodings are possible if the client defines character encoding in the request header, or if the code uses the setCharacterEncoding(String encoding) method.

If the autoRequestEncoding value is set to `true`, and the client did not specify character encoding in the request header, and the code does not include the setCharacterEncoding(String encoding) method, the Web container tries to determine the correct character encoding for the request parameters and data.

The Web container performs each step in the following list until a match is found:

- Looks at the character set (charset) in the Content-Type header.
- Attempts to map the server's locale to a character set using defined properties.
- Attempts to use the DEFAULT_CLIENT_ENCODING system property, if one is set.
- Uses the ISO-8859-1 character encoding as the default.

If the autoResponseEncoding value is set to `true`, and the client did not specify character encoding in the request header, and the code does not include the setCharacterEncoding(String encoding) method, the Web container does the following:

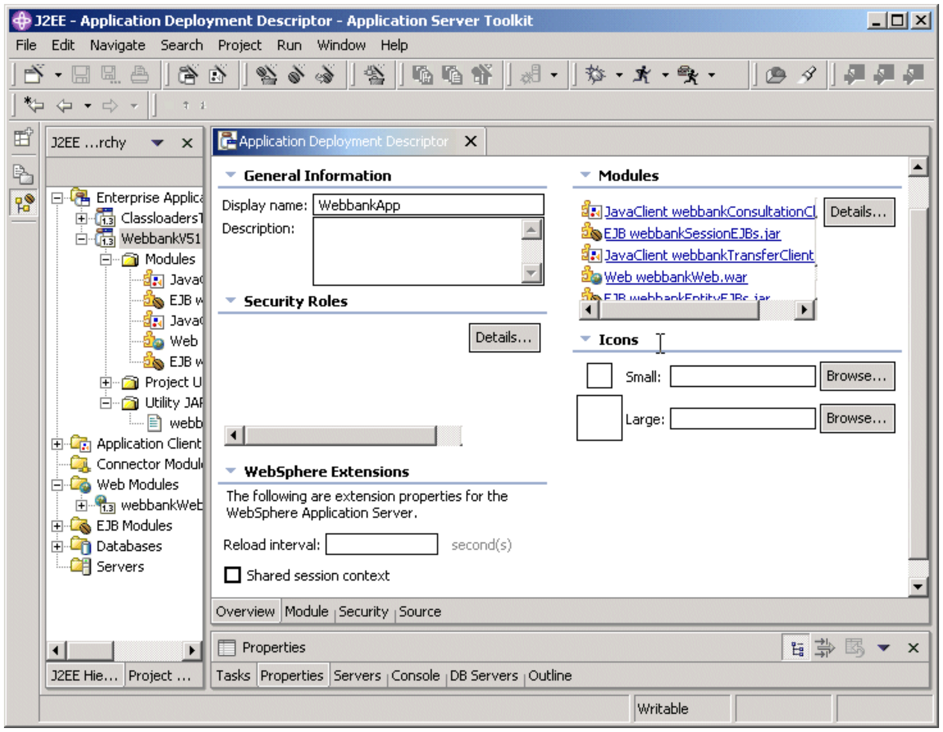
- Attempts to determine the response content type and character encoding from information in the request header.
- Uses the ISO-8859-1 character encoding as the default.

13.8 IBM EAR extensions: Sharing session context

In accordance with the servlet 2.3 API specification, the session manager supports session scoping by Web module only. Only servlets in the same Web module can access the data associated with a particular session. WebSphere Application Server provides an option that you can use to extend the scope of the session attributes to an enterprise application. Therefore, you can share session attributes across all the Web modules in an enterprise application.

This option can be set in the toolkit in the enterprise application deployment descriptor.

1. Open the deployment descriptor by double-clicking the enterprise application.
1. Check the **Shared httpsession context** option in the WebSphere Extensions section.


J2EE - Application Deployment Descriptor - Application Server Toolkit
File Edit Navigate Search Project Run Window Help
J2EE ...rchy
Application Deployment Descriptor
General Information
Security Roles
Details...
WebSphere Extensions
Modules
Details...
Icons
Browse...
Browse...
Overview Module Security Source
Properties
Tasks Properties Servers Console DB Servers Outline
J2EE Hie... Project ...
Writable

Display name: WebbankApp

Description:

JavaClient webbankConsultationCl
EJB webbankSessionEJBs.jar
JavaClient webbankTransferClient
Web webbankWeb.war
EJB webbankEntityEJBs.jar

Small:

Large:

The following are extension properties for the WebSphere Application Server.

Reload interval: second(s)

☐ Shared session context

Figure 13-11 EAR deployment descriptor

Important: To use this option, you must install all the Web modules in the enterprise application on the same server. You cannot use this option when one Web module is installed on one server and the second Web module is installed on a different server. In such split installations, applications might share session attributes across Web modules using distributed sessions, *but* session data integrity is compromised when concurrent access to a session is made in different Web modules. Sharing HTTP session context also severely restricts the use of some session management features, like time-based writes. For enterprise applications on which this option is enabled, the session management configuration set at the Web module level is ignored. Instead, the session management configuration defined at the enterprise application level is used.

13.9 Verifying the contents of an archive

As you update and save modules in the toolkit, the contents of the modules will be automatically validated and problems will be listed in the Tasks view. You can also manually invoke validation of modules by selecting any module and choosing **Run Validation** from the context menu. To verify the settings for validation, select **Window -> Preferences** and click **Validation**.

13.10 Packaging recommendations

Here are some basic rules to consider when packaging an enterprise application:

- The EJB JAR modules and Web WAR modules comprising an application should be packaged together in the same EAR module.
- When a Web module accesses an EJB module, you should not package the EJB interfaces and stubs in the WAR. Thanks to the classloading architecture, EJB stubs and interfaces are visible by default to WAR modules.
- Utility classes used by a single Web module should be placed within its WEB-INF/lib folder. This is what we have done for the struts.jar file used only by the WebbankWeb module.
- Utility classes used by multiple modules within an application should be placed at the root of the EAR file. This is what we have done for the webbankCommon.jar, which is used both by servlets and EJBs.
- Utility classes used by multiple applications can be placed on a directory referenced via a Shared Library definition.

See also 14.7, “Learning classloaders by example” on page 696 for more details on how WebSphere actually finds and loads classes.

14

Deploying applications

The previous chapter has taken you through packaging the Webbank application. In this chapter, you get to deploy it. First, we take you through setting up the right environment for the application, such as creating an application server, virtual hosts, data sources, and binding these resources to the Webbank application. Then, we take you through deploying the Webbank application. Next, we explain how to deploy the client part of the application. We also describe how to separate the static content from the dynamic content of your application. Finally, we describe how WebSphere classloaders work, as well as give you some packaging and maintenance advice.

In this chapter, we assume you are familiar with basic WebSphere administrative tasks, such as starting the administrative console.

In this chapter, you will use the Assembly Toolkit and the administrative console exclusively. All deployment tasks can be automated using command-line tools as explained in Chapter 16, “Command-line administration and scripting” on page 779.

14.1 Preparing the environment

To deploy the Webbank application, we do the following:

- Create an application server to host the application.
- Define the necessary resources, such as data sources and virtual hosts.
- Configure the WebSphere JMS provider (to support the message-driven bean at runtime).

Steps in this section are typically performed by the application deployer.

14.1.1 Creating a WEBBANK_ROOT environment variable

It is recommended that you use WebSphere environment variables rather than hard-coded paths when deploying an application. In the following steps, we assume you have declared a WEBBANK_ROOT variable, which you will use when specifying, for example, the JVM log's location.

Make sure you declare this variable at the right scope. For example, if you define this variable at the application server scope, it will only be known at that level. As long as you work with the WebSphere Application Server (base edition), this is fine. But if you later decide to use the Network Deployment edition and you create a cluster of application servers, the WEBBANK_ROOT variable will need to be defined at the node level if all members of a cluster are on the same node, and at cell level if the cluster spans multiple nodes.

Use the steps in 8.6.1, "Using variables" on page 277 to create a WEBBANK_ROOT variable with a value of, for example, F:\WebbankSample.

14.1.2 Creating the Webbank application server

When WebSphere Application Server is installed, a default server (server1) is created automatically. Although the Webbank application could be deployed to this default server, we recommend that you create a separate server for your applications. The default server is intended to run the WebSphere samples and serve as the server template.

Note: For a full discussion of application server properties, see 8.6.3, "Managing application servers" on page 283.

To create an application server:

1. Select **Servers -> Application Servers**.

2. Click the **New** button and provide the following information (as shown in Figure 14-1):

   – Node name:

     The node on which the application server will be created. If you are using the base edition, you have a single choice. If you are using the Network Deployment edition, you can choose among all the nodes in the cell you are currently managing.

   – Server name:

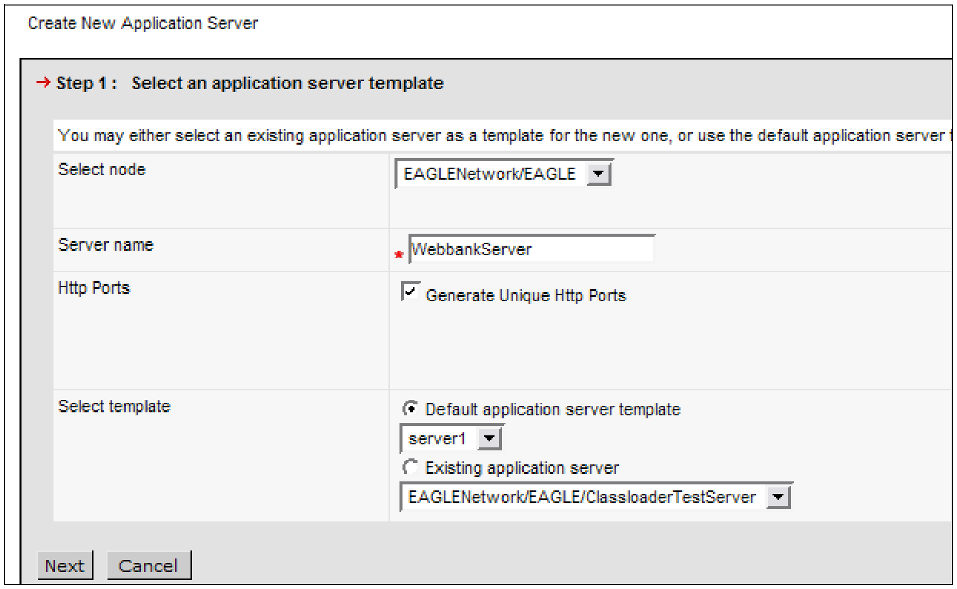
     The application server name, such as WebbankServer.

   – Generate HTTP ports:

     Check this if you want WebSphere to generate unique ports for this server. This ensures the ports used by this server will not conflict with another server started on the same node.

   – Server Template:

     A server definition you inherit properties from. This server definition file will be used as the base to define the new server. You may want to predefine a server template with customized settings and reuse it as the base for all your application servers. You may also choose an existing application server as a template.



Create New Application Server

→ Step 1 : Select an application server template

You may either select an existing application server as a template for the new one, or use the default application server

Select node	EAGLENetwork/EAGLE
Server name	* WebbankServer
Http Ports	✓ Generate Unique Http Ports
Select template	● Default application server template server1 ○ Existing application server EAGLENetwork/EAGLE/ClassloaderTestServer

Figure 14-1 Creating the Webbank application server

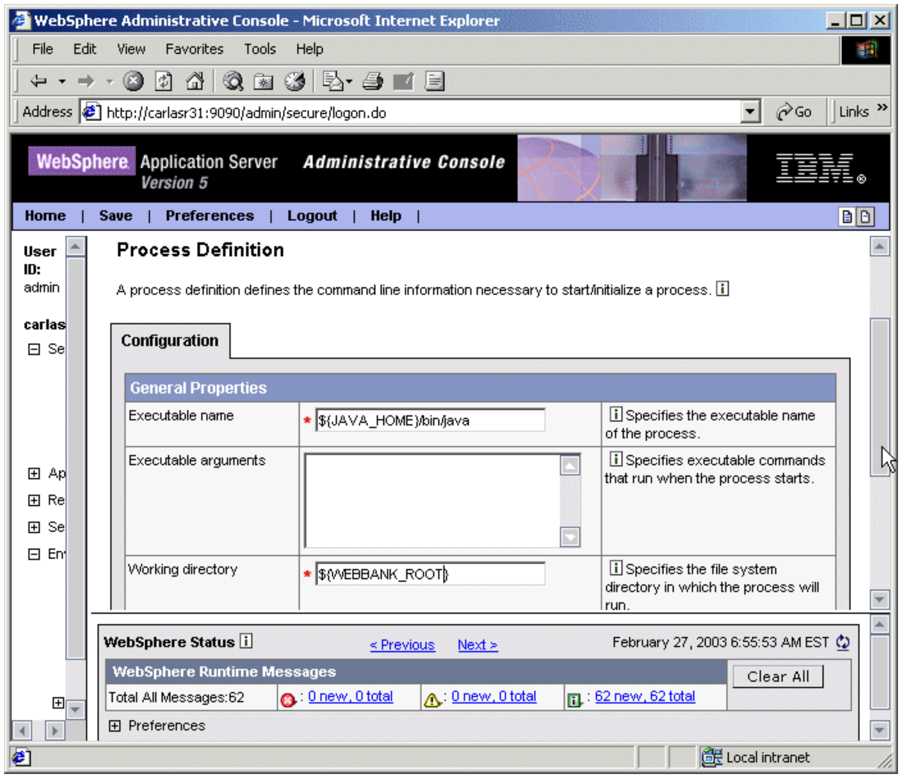
Tip: We recommend that you use templates to generate new server definition. Simply define an application server, with all the default settings your environment may require, and use it as a template for other application servers.

3. Click **Next.**
4. The next window lets you review the current application server settings. From there, you can go back to the previous page or confirm the settings and complete the application server creation by clicking **Finish**.

Changing the process definition

The next thing we want to do is to change the process working directory. This directory is the relative root for searching files. For example, if you do a File.open("foo.gif"), foo.gif must be present in the working directory to be found. This directory will be created by WebSphere if it does not exist. We recommend that you create a specific working directory for each application server.

1. Select the server, **WebbankServer**, you just created.
2. Select **Process Definition** from the Additional Properties table.
3. Change the default working directory from ${USER_INSTALL_ROOT} to ${WEBBANK_ROOT}/workingDir.



Process Definition

A process definition defines the command line information necessary to start/initialize a process.

Configuration

General Properties

Executable name	* ${JAVA_HOME}/bin/java	Specifies the executable name of the process.
Executable arguments		Specifies executable commands that run when the process starts.
Working directory	* ${WEBBANK_ROOT}	Specifies the file system directory in which the process will run.

WebSphere Status — February 27, 2003 6:55:53 AM EST

WebSphere Runtime Messages

Total All Messages:62 — 0 new, 0 total — 0 new, 0 total — 62 new, 62 total

WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://carlasr31:9090/admin/secure/logon.do
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help
User ID: admin
< Previous Next >
Clear All
Preferences
Local intranet

Figure 14-2 Change the working directory for the server

4. Click **OK**.

Note: The working directory won’t be created if you use a composed path, such as F:/WebbankSample/workingDir. If you wish to use such a path, make sure you create it before starting the application server, or the startup sequence will fail.

Changing the logging and tracing options

Next, we want to customize the logging and tracing properties for the new server. These properties are discussed in detail in Chapter 15, “Troubleshooting” on page 715. There are two ways to access the logging and tracing properties for an application server:

- You can select **Troubleshooting -> Logs and Trace** in the navigation bar, then select a server.

- Or, you can select **Servers -> Application Servers**, select a server, and then select **Logging and Tracing** from the Additional Properties table.

Since we are already in the server configuration windows, we will take the second route.

1. Select **Logging and Tracing** from the Additional Properties table of the server.
2. Select **Process Logs**.

   This allows you to change the JVM standard output and error file properties. Both are rotating files. You can choose to save the current file and create a new one, either when it reaches a certain size, or at a specific moment during the day. You may also choose to disable the output of calls to System.out.print() or System.err.print().

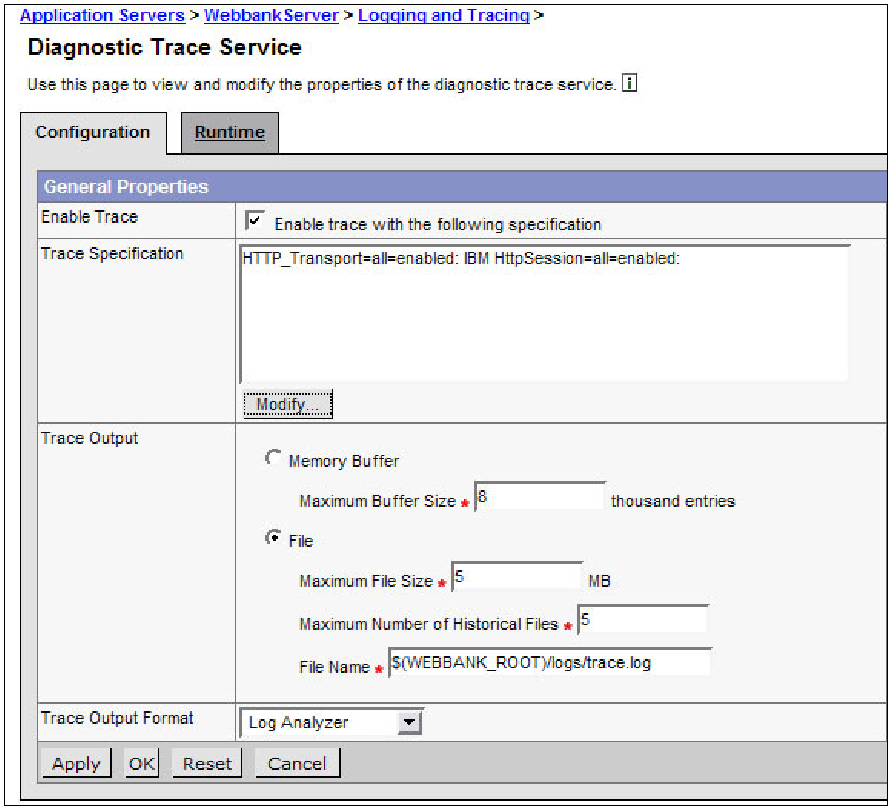
   We recommend that you specify a new file name, using an environment variable to specify it, such as:

   ```
   ${WEBBANK_ROOT}/logs/stdout.txt and ${WEBBANK_ROOT}/logs/stderr.txt
   ```

   Click **OK**.
3. Select **Diagnostic Trace.**

   Each component of the WebSphere Application Server is enabled for tracing via the JRas interface. This trace can be dynamically changed while the process is running from the Runtime tab or added to the application server definition from the Configuration tab. As shown in Figure 14-3 on page 653, the trace output can be either directed to memory or to a rotating trace file.

   Change the trace output file name so the trace is stored in a specific location for the server using the WEBBANK_ROOT variable and select the **Log Analyzer** format.


Application Servers > WebbankServer > Logging and Tracing >

Diagnostic Trace Service

Use this page to view and modify the properties of the diagnostic trace service.

General Properties	
Enable Trace	[✓] Enable trace with the following specification
Trace Specification	HTTP_Transport=all=enabled: IBM HttpSession=all=enabled:
Trace Output	( ) Memory Buffer Maximum Buffer Size * 8 thousand entries (●) File Maximum File Size * 5 MB Maximum Number of Historical Files * 5 File Name * $(WEBBANK_ROOT)/logs/trace.log
Trace Output Format	Log Analyzer

Figure 14-3 Specifying diagnostic trace service options

14.1.3 Changing the WebbankServer HTTP transport port

When we created the application server, we let the system define a default HTTP port for the Web container. However, in the following sections, we assume this port has been set to 9085. Therefore, you should change it by following these steps:

1. Select **Servers -> Application Servers**.
2. Select the **WebbankServer**.
3. Select **Web Container** in the Additional Properties table.
4. Select **HTTP transports** in the Additional Properties table.
5. Find the entry with the following characteristics (most likely the first entry):
   – Host: *
   – SSL Enabled: false

6. If the port is not 9085, open the transport definitions by clicking the host (*) and modify the current port setting to be 9085. Click **OK**.
7. Save the configuration.

14.1.4 Defining the Webbank virtual host

Web modules need to be bound to a specific virtual host. For our sample, we chose to bind the Webbank Web module to a specific virtual host called webbank_vhost. This virtual host has the following host aliases:

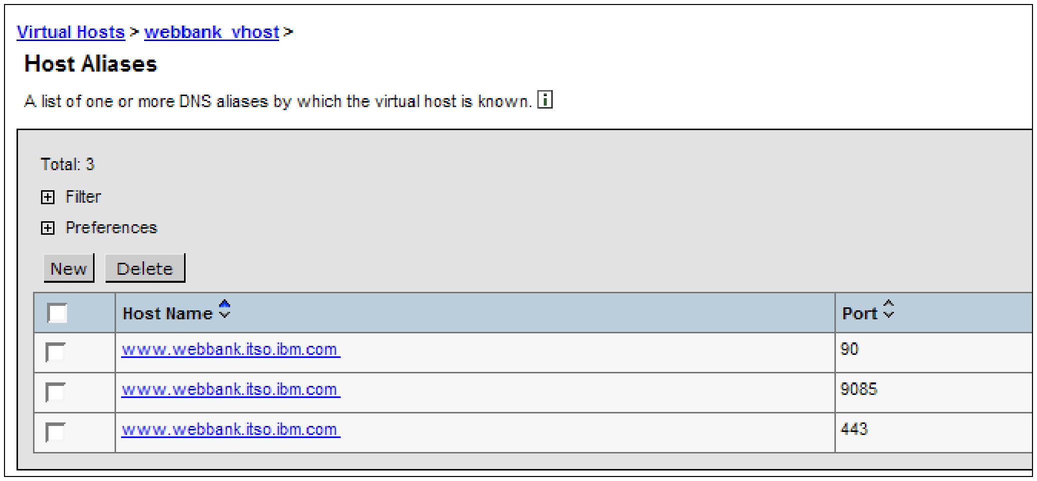
- www.webbank.itso.ibm.com:90
- www.webbank.itso.ibm.com:9085
- www.webbank.itso.ibm.com:443 (for SSL access)

Any request starting with `<webbank_vhost_host_alias>/webbank`, such as `http://www.webbank.itso.ibm.com:90/webbank/TransferServlet`, is served by the Webbank Web application. Details about virtual hosts and how they are used by the Web server plug-in can be found in 8.6.5, "Managing virtual hosts" on page 309.

Tip: You can restrict the list of hosts that can be used to access the Webbank Web application by removing this host from the virtual host definition. Say you want to prevent people from directly accessing the Webbank application via the WebSphere internal HTTP server by invoking `http://www.webbank.itso.ibm.com:9085/webbank` (in other words, you want to force all requests to go through the Web server plug-in). You can achieve this by removing the `www.webbank.itso.ibm.com:9085` entry from the virtual host aliases list. The application server will still be listening on that port, and that port will still be used in the plugin configuration file. However, the application server will prevent any direct call to this port.

To create the webbank_vhost virtual host, do the following:

1. Select the **Environment -> Virtual Hosts** entry in the navigation pane and click **New** button in the right pane.
2. Supply the virtual host name, that is "webbank_vhost".
3. Click **Apply**.
4. Select Host Aliases in the **Additional Properties** table. Add the three aliases shown in Figure 14-4 on page 655 by clicking **New**, entering the values, and clicking **OK**.


Virtual Hosts > webbank_vhost >

Host Aliases

A list of one or more DNS aliases by which the virtual host is known.

Total: 3

⊞ Filter

⊞ Preferences

	Host Name	Port
☐	www.webbank.itso.ibm.com	90
☐	www.webbank.itso.ibm.com	9085
☐	www.webbank.itso.ibm.com	443

Figure 14-4 Webbank virtual host aliases

5. Click **OK**.
6. Save the configuration.

14.1.5 Creating the virtual host for IBM HTTP Server (or Apache)

Now that we have defined a webbank_vhost virtual host, we need to configure the Web server to serve the host aliases defined in the virtual host. We also need to configure the Web server to listen on port 90. The steps below are valid for both the IBM HTTP Server and Apache 2.0.

Configuring listening ports

For this sample, we have chosen to use port 90. This is mainly for the sake of demonstrating how it can be done. You can just as well use the default port (80). Just remember that the port you use must be defined in the virtual host alias list. You can use the Listen directive to tell the Web server to listen to requests on a specific port:

1. Make a backup of the <IHS_HOME>\conf\httpd.conf file.
1. Start your favorite editor and edit this file.
2. Search for the line containing the `# Listen` string, and edit it as follows:

```
Listen 90
```

3. Save the httpd.conf file.

Configuring virtual hosting

Creating virtual hosts is done using the VirtualHost directive, like this:

```
<VirtualHost www.webbank.itso.ibm.com:90>
   ServerAdmin webmaster@itsowebbank.com
   ServerName www.webbank.itso.ibm.com
   DocumentRoot "F:/IBMHTTPServer/htdocs/webbank"
   ErrorLog logs/webbankHost-error.log
   TransferLog logs/webbankHost-access.log
</VirtualHost>
```

If you want to have multiple virtual hosts for the same IP address, you must use the NameVirtualHost directive, as shown in Example 14-1.

Example 14-1 Using the NameVirtualHost and VirtualHost directives

```
NameVirtualHost 9.100.11.41:90

<VirtualHost itso_server:90>
   ServerAdmin webmaster@itso_server.com
   ServerName itso_server
   DocumentRoot "F:/IBMHTTPServer/htdocs/itso_server"
   ErrorLog logs/itso_server-error.log
   TransferLog logs/itso_server-access.log
</VirtualHost>

<VirtualHost www.webbank.itso.ibm.com:90>
   ServerAdmin webmaster@webbank.itso.ibm.com
   ServerName www.webbank.itso.ibm.com
   DocumentRoot "F:/IBMHTTPServer/htdocs/webbank"
   ErrorLog logs/webbankHost-error.log
   TransferLog logs/webbankHost-access.log
</VirtualHost>
```

The `www.webbank.itso.ibm.com` and the `itso_server` hosts have the same IP address, that is 9.100.11.41. We have set this by inserting the following line in the machine `hosts` file (located in %windir%\system32\drivers\etc or in /etc on UNIX systems):

```
9.100.11.41 www.webbank.itso.ibm.com itso_server
```

In a real-life environment, this would probably be achieved by creating aliases at the DNS level. In any event, you must be able to ping the host you have defined, using commands such as **`ping www.webbank.itso.ibm.com`**.

As you can see in Example 14-1, each virtual host has a different document root. Make sure that the directory you specify exists before you start the HTTP server. We recommend that you place an index.html file at the document root stating which virtual host is being called. This lets you easily test which virtual host is being used.

You must restart the IBM HTTP Server to apply these changes. If you are running a Windows system, we recommend that you try to start the server by running **`apache.exe`** from the command line rather than from the Services window. This allows you to spot error messages thrown at server startup.

If your virtual hosts are correctly configured, invoking `http://itso_server:90` or `http://www.webbank.itso.ibm.com:90` will return different HTML pages.

14.1.6 Creating a DB2 JDBC provider and data source

The Webbank application uses a relational database, via entity beans, to store account information. To access this database, you have to define a data source, which you then associate with the entity beans. For this sample, we have chosen to create a specific JDBC provider for the Webbank application. Additionally, you must create J2C authentication data for the data source, and finally associate a data source with the Webbank JDBC provider. For detailed information about JDBC providers and data sources, refer to 12.1, “JDBC resources” on page 550.

Tip: If you are installing the sample application, you have the option of using Cloudscape instead of DB2. The steps for Cloudscape are in Appendix D, “Webbank application overview” on page 917.

Creating the Webbank UDB provider

The following steps take you through the creation of a JDBC provider targeting a DB2 database. To create a JDBC provider from the administrative console:

1. Expand the **Resources** entry and select the **JDBC Providers** entry.
2. Select the scope of this resource. If you are deploying in a Base edition server, it is sufficient to create the data source at the server level. Otherwise, you should define it at the node or cell level (for example, to be able to share the definition across n servers in a cluster). To change this, select the server you are deploying to in the scopes list and click **Apply**.
3. Click the **New** button.
4. In the list of supported JDBC providers, select the **DB2 Universal JDBC Driver Provider** entry (the preferred choice for UDB V8); you can also use another JDBC provider if you have decided to deploy the Webbank application on a different database.
5. Click **Apply**.

6. Specify the following information, as shown in Figure 14-5 on page 659:

   – Name

     The JDBC provider name, such as “WebbankUDBProvider”. If you are defining the JDBC provider to service one database, for clarity, include the name of the target database in this name.

   – Description

     An optional description of the JDBC provider.

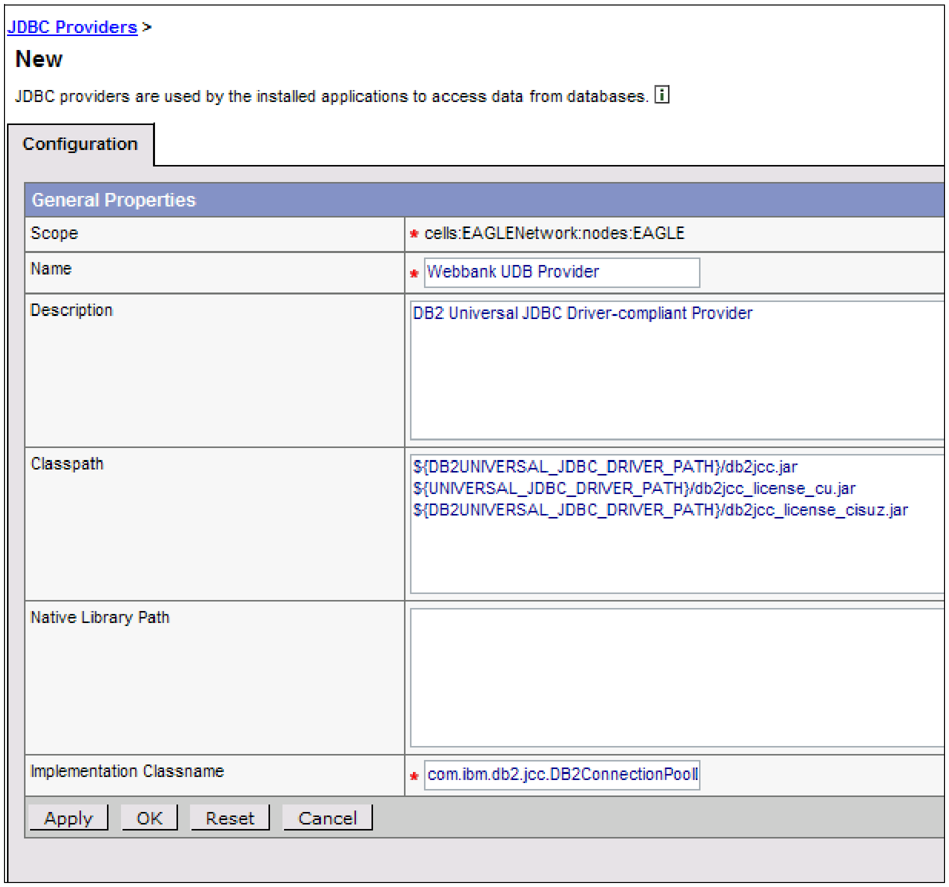
   – Classpath

     The path to the JAR/ZIP file that contains the JDBC provider code, for example, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip. We encourage you to use variables rather than hard-code the path to the db2java.zip or classes12.zip file.

Important: Make sure that *all* the variables used in the classpath, such as the DB2UNIVERSAL_JDBC_DRIVER_PATH variable, have been correctly set. By default, those variables are defined, but not set. This can be done by selecting **Environment -> Manage WebSphere Variables**. Also, make sure that the variable has been set *at the same scope or higher* than the JDBC provider.

   – Implementation class

     The Java class that provides the JDBC service, such as `com.ibm.db2.jcc.DB2ConnectionPoolDataSource` or `oracle.jdbc.pool.OracleConnectionPoolDataSource`. This field is automatically completed for known JDBC providers.


JDBC Providers >

New

JDBC providers are used by the installed applications to access data from databases.

Configuration

General Properties

Scope	* cells:EAGLENetwork:nodes:EAGLE
Name	* Webbank UDB Provider
Description	DB2 Universal JDBC Driver-compliant Provider
Classpath	${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar
Native Library Path	
Implementation Classname	* com.ibm.db2.jcc.DB2ConnectionPoolI

Figure 14-5 Webbank JDBC provider properties

7. Click **OK**.

Note: Although not required for this sample, note that if you need two-phase commit support for a data source, you should select a driver that contains the “XA” keyword. We recommend that you indicate this is a JTA-enabled provider by including the JTA keyword in the provider name, such as JTAWebbankDB2Provider.

Configuring J2C authentication data

Version 5.0 data sources use J2C security. Data source connection information (user ID/password) is not specified at the data source level anymore. You must define a J2C authentication alias. To create a J2C authentication alias:

1. Select **Security -> JAAS Configuration -> J2C Authentication Data**.

2. Click **New**, and specify the following information to create the authentication data. Once completed, the authentication information should be similar to Figure 14-6.

   – Alias

     The name of the security information entry, such as `webbank_auth_alias`.

   – User ID

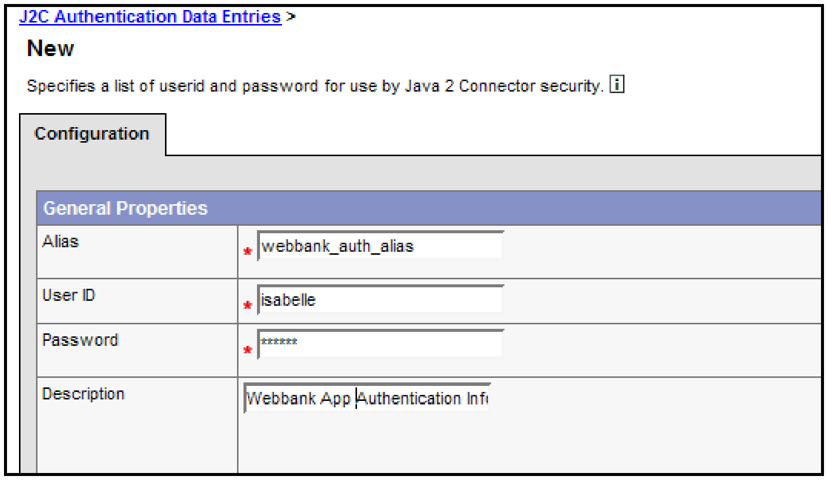
     The user ID for this alias.

   – Password

     The password for this alias.

   – Description

     An optional (but recommended) description of the alias.



J2C Authentication Data Entries >

New

Specifies a list of userid and password for use by Java 2 Connector security.

Configuration

General Properties

Alias	* webbank_auth_alias
User ID	* isabelle
Password	* ******
Description	Webbank App Authentication Inf

Figure 14-6 Specifying the Webbank data source authentication data

3. Click **OK**.

Creating the Webbank data source

The next step is to create a data source that uses this JDBC provider. To create a data source, you need to do the following:

1. Select **Resources -> JDBC Providers**.
2. Select the **WebbankUDBProvider** JDBC provider and select its corresponding **Data Sources** entry.
3. Click **New**.

4. As shown in Figure 14-7 on page 662, the following properties can be specified for a data source:

   – Name

     The data source name, which must be unique in the administrative domain (cell). It is recommended that you use a value indicating the name of the database this data source is targeting, such as “WebbankDS”.

   – JNDI name

     The name by which applications access this data source. If not specified, the JNDI name defaults to the data source name prefixed with “jdbc/”. If you specify a JNDI name for a data source, we recommend that you prefix it with “jdbc/” to respect the naming conventions of the J2EE specification.

     For the Webbank application, this field can be set to “`webbank/jdbc/webbank`”. This value can be changed at any time after the data source has been created.

   – Description

     An optional (but recommended) description of this data source.

   – Container-managed persistence

     Check this option to indicate that the Webbank data source is used for persisting the application entity beans.

   – Category

     A keyword, such as Applications or Samples, used to classify data sources. The list of data sources can be sorted using that field.

   – Statement Cache Size

     For better efficiency, WebSphere can maintain a cache of results of the ps.prepareStatement() call. Use this setting to set the size of this cache.

   – Datasource helper name

     The name of the class used by the Relational Resource Adapter at runtime to implement the functionality of a specific database driver. This field is set automatically based on the JDBC provider type (for example, `com.ibm.websphere.rsadapter.OracleDataStoreHelper` for non-XA Oracle access or `com.ibm.websphere.rsadapter.DB2DataStoreHelper` for non-XA DB2 access).

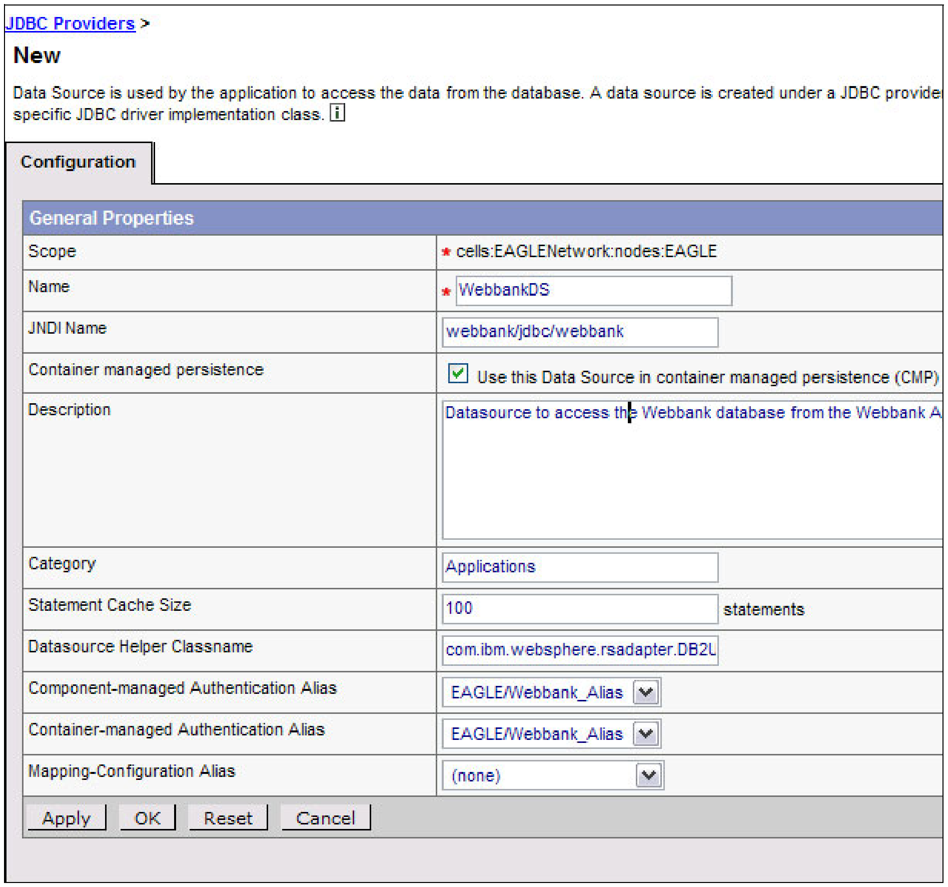
   – Authentication Aliases

     The J2C aliases used when connecting to this data source, either at the container or application level depending on the res-auth tag for your application. If res-auth is set to `Application`, and you do not specify a user ID when obtaining a connection (that is, when datasource.getConnection()

is called), the component-managed authentication alias is used. If res-auth is set to `Container`, the container-managed alias is used.

– Mapping-Configuration Alias

Select from defined JAAS application login configurations.



JDBC Providers >

New

Data Source is used by the application to access the data from the database. A data source is created under a JDBC provide specific JDBC driver implementation class.

Configuration

General Properties

Scope	* cells:EAGLENetwork:nodes:EAGLE
Name	* WebbankDS
JNDI Name	webbank/jdbc/webbank
Container managed persistence	✓ Use this Data Source in container managed persistence (CMP)
Description	Datasource to access the Webbank database from the Webbank A
Category	Applications
Statement Cache Size	100 statements
Datasource Helper Classname	com.ibm.websphere.rsadapter.DB2U
Component-managed Authentication Alias	EAGLE/Webbank_Alias
Container-managed Authentication Alias	EAGLE/Webbank_Alias
Mapping-Configuration Alias	(none)

Figure 14-7 Webbank data source properties

5. Click **Apply**.
6. Select **Custom Properties** in the Additional Properties table to edit a list of properties specific to the database driver.

Note: For DB2, you need at least to specify the database name (such as Webbank). For Oracle, you would have to provide the driver type (thin/oci8), the database URL, or server name/listening port.

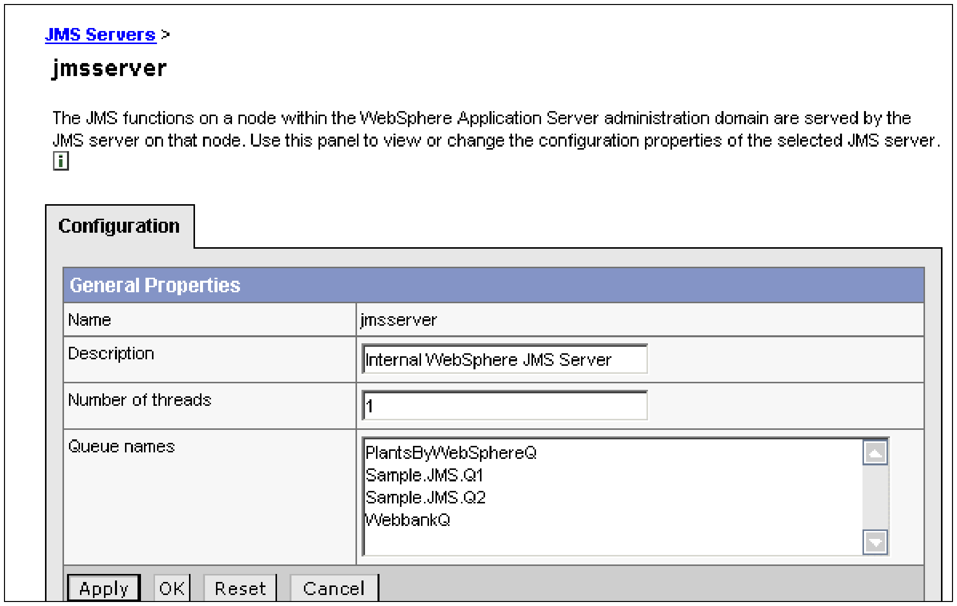
7. Select **databaseName**.
8. Change the value to WEBBANK.
9. For the Universal UDB driver, you must also specify the *driver type* (2 or 4) as well as the target server name custom properties. This is done in the same manner as setting the database name. For this example, we used driver type `4` (the default). The server name is set to the database server, in this case, `localhost`.
10. Click **OK**.
11. Test the connection by selecting the data source and clicking the **Test Connection** button.

14.1.7 Configuring the WebSphere JMS provider

We will now go through the various steps necessary to run the Transfer message-driven bean.

The transferMDB has been defined to listen to a JMS queue. This JMS queue needs to be available somewhere. You can use any JMS-compliant provider for this queue. In the sample, we have decided to use the embedded WebSphere JMS provider, a default JMS provider that comes with WebSphere Application Server V5. Follow these steps to add the Webbank queue to the default JMS provider:

1. In the administrative console, select **Servers -> JMS Servers**.
2. Select **jmsserver**.
3. In the list of queues, add the `WebbankQ` queue to the list of queue names, as shown in Figure 14-8 on page 664. Just type the name in the window and click **OK**.



jmsserver

The JMS functions on a node within the WebSphere Application Server administration domain are served by the JMS server on that node. Use this panel to view or change the configuration properties of the selected JMS server.

General Properties

Name	jmsserver
Description	Internal WebSphere JMS Server
Number of threads	1
Queue names	PlantsByWebSphereQ Sample.JMS.Q1 Sample.JMS.Q2 WebbankQ

Figure 14-8 Adding the WebbankQ queue name to the JMS server definition

4. Click **OK**.
5. Save the configuration.

Important: Be sure to (re)start this server before running the Webbank application!

14.1.8 Creating JMS resources

The next step is to create the necessary JMS resources, the same way you created JDBC resources to be able to access relational data. Two resources must be created: a queue connection factory and a queue.

JMS resources can be created using the administrative console from the Resources entry. You have three choices for JMS resources:

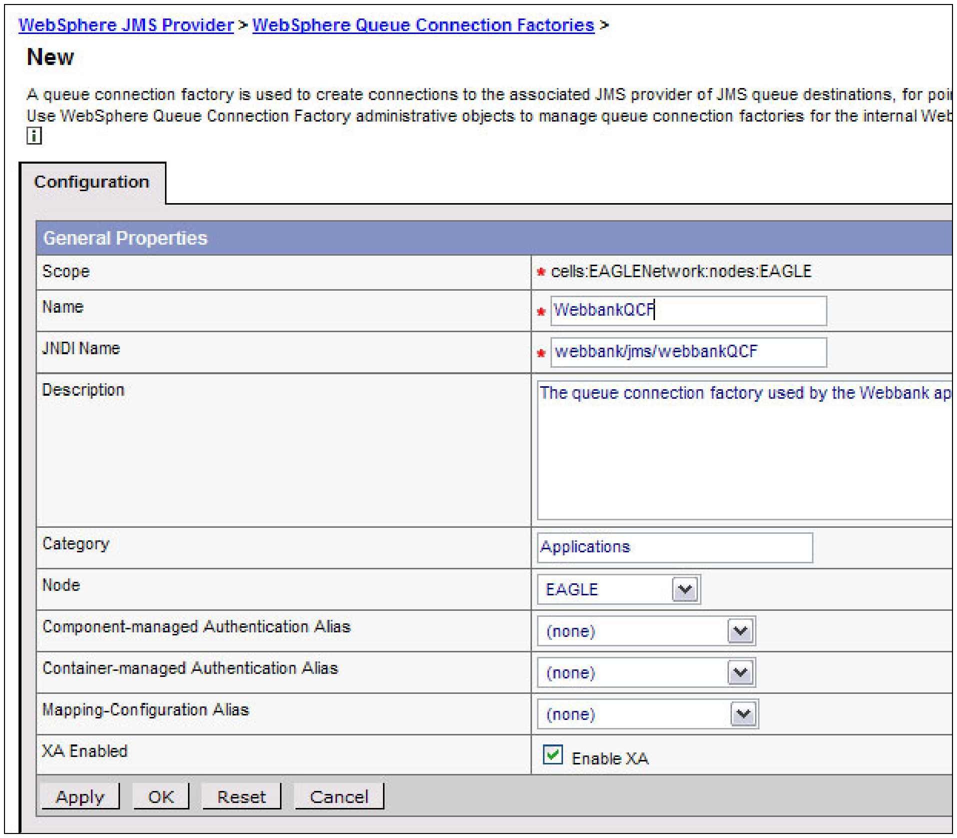
- Generic JMS Providers: This option must be used if your JMS provider is a generic J2EE provider. You must specify all the information to access your provider.
- WebSphere JMS Provider: This option has been pre-configured to access the internal JMS provider (this is what we need).

- WebSphere MQ JMS Provider: This is specific to WebSphere MQ (previously called MQSeries).

In this sample, the WebSphere JMS Provider installed with WebSphere Application Server is used.

To create a WebSphere queue connection factory:

1. Select **Resources -> WebSphere JMS Provider**.
2. Select **WebSphere Queue Connection Factories** in the Additional Properties table.
3. Click **New**, and provide the following information:
   - Name: The queue connection factory name, such as `WebbankQCF`.
   - JNDI name: `webbank/jms/webbankQCF`. As usual, we recommend that you create a subcontext for the application and another one for the resource type, here `jms`.
   - Description: Optional description of the resource.
   - Category: An optional string used to classify resources.
   - Node: The node where the JMS provider that supports this resource runs.
   - Authentication aliases: J2C authentication data for this resource.
   - XA-enabled: Whether this resource can be participate in two-phase commit transactions.


WebSphere JMS Provider > WebSphere Queue Connection Factories >

New

A queue connection factory is used to create connections to the associated JMS provider of JMS queue destinations, for poi
Use WebSphere Queue Connection Factory administrative objects to manage queue connection factories for the internal Web

Configuration

General Properties	
Scope	* cells:EAGLENetwork:nodes:EAGLE
Name	* WebbankQCF
JNDI Name	* webbank/jms/webbankQCF
Description	The queue connection factory used by the Webbank ap
Category	Applications
Node	EAGLE
Component-managed Authentication Alias	(none)
Container-managed Authentication Alias	(none)
Mapping-Configuration Alias	(none)
XA Enabled	✓ Enable XA

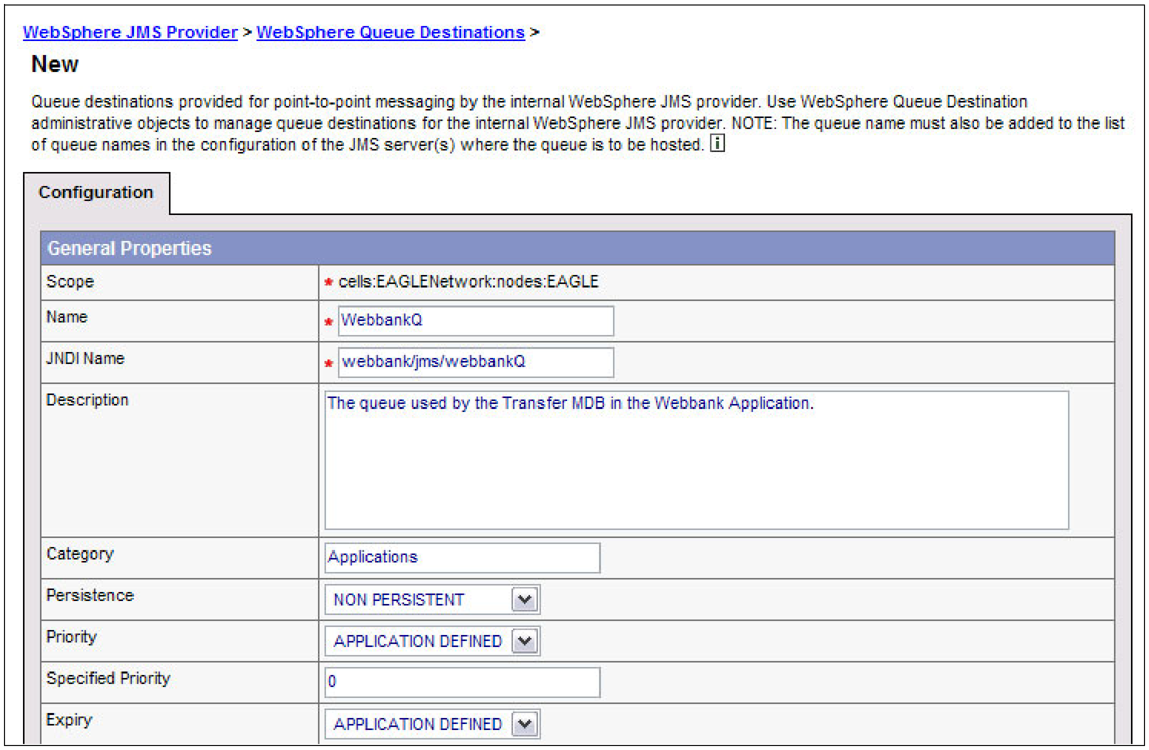
Figure 14-9 Creating the Webbank queue connection factory

4. Click **OK**.

To create a WebSphere queue destination, you must:

1. Select the **Resources -> WebSphere JMS Provider** entry.
2. Select **WebSphere Queue Destinations** In the Additional Properties table.
3. Click **New** and provide the following information, as shown in Figure 14-10 on page 667.
   – Name: `WebbankQ`. This name must exactly match the queue name you have added to the internal JMS Server queue names list in 14.1.7, “Configuring the WebSphere JMS provider” on page 663. Note that this name is case-sensitive. WebSphere uses the name to link the queue definition to the actual queue running in the internal JMS server.
   – JNDI name: `webbank/jms/webbankQ`.

- Description: An optional description of the resource.
- Category: An optional string used to classify resources.



WebSphere JMS Provider > WebSphere Queue Destinations >

New

Queue destinations provided for point-to-point messaging by the internal WebSphere JMS provider. Use WebSphere Queue Destination administrative objects to manage queue destinations for the internal WebSphere JMS provider. NOTE: The queue name must also be added to the list of queue names in the configuration of the JMS server(s) where the queue is to be hosted.

Configuration

General Properties	
Scope	* cells:EAGLENetwork:nodes:EAGLE
Name	* WebbankQ
JNDI Name	* webbank/jms/webbankQ
Description	The queue used by the Transfer MDB in the Webbank Application.
Category	Applications
Persistence	NON PERSISTENT
Priority	APPLICATION DEFINED
Specified Priority	0
Expiry	APPLICATION DEFINED

Figure 14-10 Creating the Webbank queue destination

4. Click **Apply**.

Refer to 11.8.4, “Configuring JMS resources” on page 509 for additional details on queue connection factories and queue destinations settings.

14.1.9 Creating the JMS listener port

The last step is to create a JMS listener port for the Webbank server. This listener port uses the JMS resources we just defined. To define a listener port:

1. Select **Servers -> Applications Servers**.
2. Select the **WebbankServer**.
3. Select **Message Listener Service** in the Additional Properties table.
4. Select **Listener Ports**.

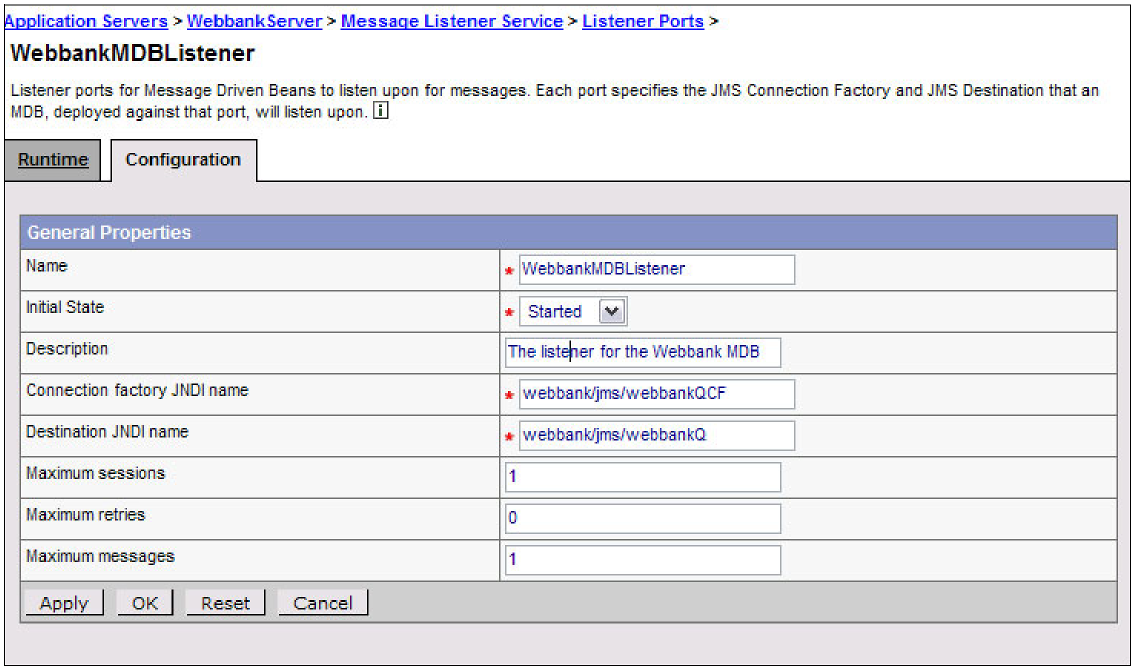
5. Click **New**, and provide the following information:

   – Name: WebbankMDBListener.

   – Initial State: Whether the listener should be started or not when the application server is started.

   – Connection factory JNDI name: `webbank/jms/webbankQCF` (this name is case-sensitive).

   – Queue destination JNDI name: `webbank/jms/webbankQ` (this name is case-sensitive).



Application Servers > WebbankServer > Message Listener Service > Listener Ports >

WebbankMDBListener

Listener ports for Message Driven Beans to listen upon for messages. Each port specifies the JMS Connection Factory and JMS Destination that an MDB, deployed against that port, will listen upon.

Name	* WebbankMDBListener
Initial State	* Started
Description	The listener for the Webbank MDB
Connection factory JNDI name	* webbank/jms/webbankQCF
Destination JNDI name	* webbank/jms/webbankQ
Maximum sessions	1
Maximum retries	0
Maximum messages	1

Figure 14-11 Create the Webbank JMS listener port

6. Click **OK**.
7. Save the configuration.

The WebSphere Application Server is now fully configured to host the Webbank application.

14.2 Setting application bindings

At packaging time, you created references to resources. Prior to deploying the application, you need to bind these references to the actual resources, such as JDBC data sources, created from the administrative console. This needs to be done for EJB references and resource references. You also need to define the enterprise bean's JNDI names, and eventually security roles.

Bindings can be defined at development or deployment time. Most probably, developers will deliver a pre-configured EAR file which will then be modified at deployment time by the deployment team. Developers use a tool like WebSphere Studio to define bindings, while deployers use the Assembly Toolkit.

In the following sections, we use the Assembly Toolkit in the Application Server Toolkit to create the binding definitions. If you are not familiar with the Application Server Toolkit, please see Appendix B, "Application Server Toolkit" on page 879.

All binding definitions are stored in the ibm-xxx-bnd.xmi files, where xxx can be ejb-jar, web, application, or application-client.

In the next steps, you define the following bindings:

- EJB JNDI names
- Data source for entity beans
- EJB references
- Virtual_host bindings for Web modules
- Message-driven bean listener port

All sections below assume that you have started the Assembly Toolkit and opened the Webbank application EAR file (WebbankV51.ear). Instructions for finding a completed WebbankV51.ear are in Appendix E, "Additional material" on page 929.

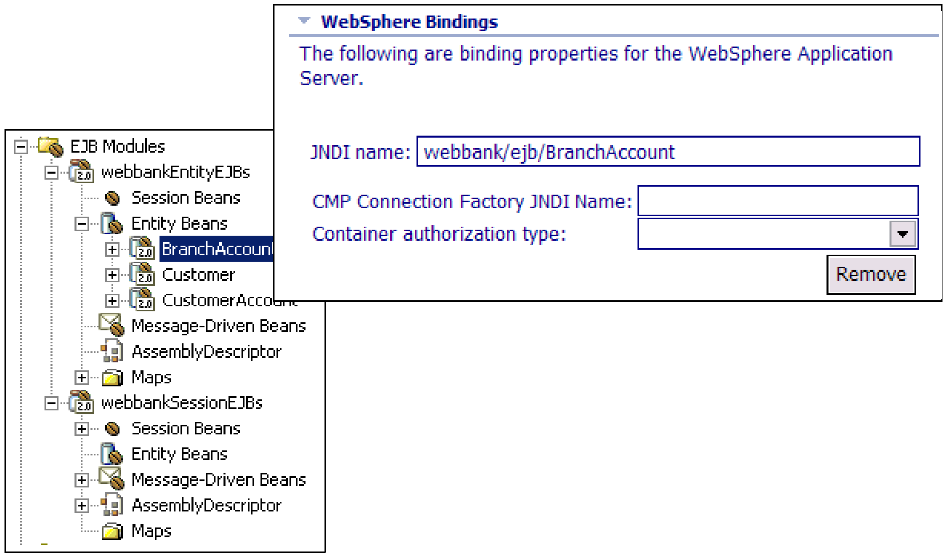
14.2.1 Defining EJB JNDI names

For each enterprise bean, you must specify a JNDI name. This name is used to bind an entry in the global JNDI name space for the EJB home object. The bind happens automatically when the application server starts.

All the Webbank enterprise beans are declared to be bound in the "webbank/" subcontext. For clarity, we recommend that you place all enterprise bean JNDI names for an application in a separate subcontext. You can find the JNDI names for each Webbank EJB in Table 14-1 on page 670. Use this table and the instructions below to define a JNDI name for each Webbank enterprise bean:

1. In the J2EE hierarchy view, go to the EJB Modules section.

2. Expand the **webbankEntityEJBs** section and the **Entity Beans** section until you see all the beans.
3. Double-click the **BranchAccount** bean. The EJB deployment descriptor editor opens to the Beans page.
4. Look for the WebSphere Bindings section in the editor.
5. In the JNDI name field, enter `webbank/ejb/BranchAccount`.



The following are binding properties for the WebSphere Application Server.

JNDI name: webbank/ejb/BranchAccount

CMP Connection Factory JNDI Name:

Container authorization type:

Figure 14-12 Defining EJB JNDI names

6. Repeat these steps for each of the enterprise beans in the same EJB module (that is, Customer and CustomerAccount). Use Table 14-1 below for correct EJB names.
7. Save the deployment descriptor.
8. Repeat all the steps for the WebbankSessionEJBs module.

Table 14-1 Webbank enterprise bean JNDI names

EJB Name	**JNDI Name**
TransferStateful session bean	webbank/ejb/TransferStateful
TransferStateless session bean	webbank/ejb/TransferStateless
Consultation session bean	webbank/ejb/Consultation
BranchAccount entity bean	webbank/ejb/BranchAccount

EJB Name	JNDI Name
CustomerAccount entity bean	webbank/ejb/CustomerAccount
Customer entity bean	webbank/ejb/Customer

14.2.2 Defining data sources for entity beans

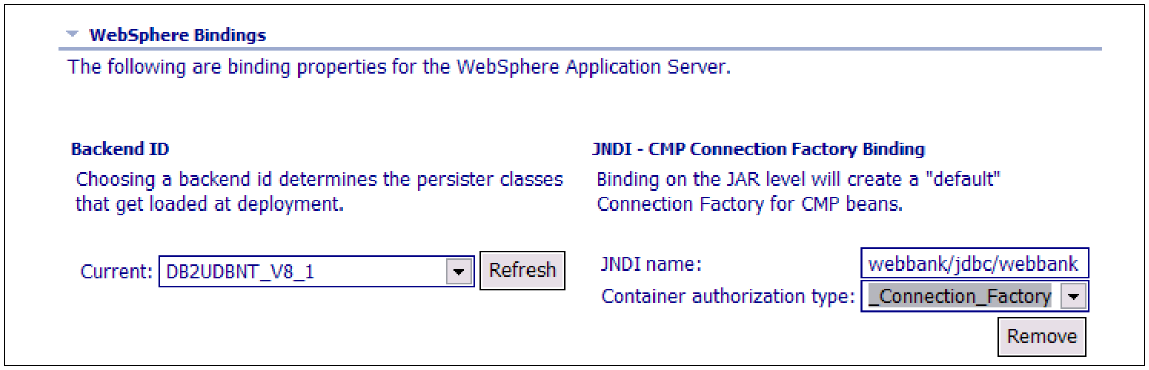
Entity beans in our application are container-managed EJBs. The EJB container handles the persistence of the EJB attributes in the underlying persistent store. You must specify which data store will be used. This is done by binding an EJB module or an individual EJB to a data source. If you bind the EJB module to a data source, all EJBs in that module use the same data source for persistence. If you specify the data source at the EJB level, then this data source is used instead.

For our sample, we chose to specify the data source at the EJB module level. This data source (or JDBC resource) has already been created and its JNDI name has been set to "`webbank/jdbc/webbank`".

To bind the Webbank Entitities EJB module to this data source, follow these steps:

1. In the J2EE view, double-click the **WebbankEntityEJBs** module to open its deployment descriptor. The editor should open to the overview.
2. Find the WebSphere bindings section.
3. In the JNDI name field, enter `webbank/jdbc/webbank`.
4. Specify whether the authentication is handled at the container or application level.
5. Save the deployment descriptor.

Tip: A same EJB JAR can contain database mappings for multiple databases. For example, the Webbank sample contains EJB generated code for UDB Version 7, UDB version 8, and Cloudscape 5. You can set which backend will be used at runtime in the WebSphere bindings section, as shown in Figure 14-13 below. This choice can also be overriden at deployment time.


WebSphere Bindings

The following are binding properties for the WebSphere Application Server.

Backend ID

Choosing a backend id determines the persister classes that get loaded at deployment.

Current: DB2UDBNT_V8_1

JNDI - CMP Connection Factory Binding

Binding on the JAR level will create a "default" Connection Factory for CMP beans.

JNDI name: webbank/jdbc/webbank

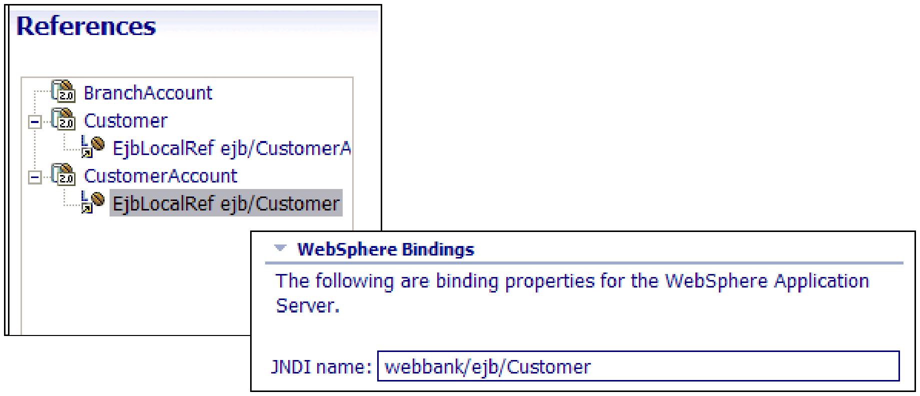
Container authorization type: _Connection_Factory

Figure 14-13 Specifying the default CMP datasource for the entity EJBs

14.2.3 Binding EJB and resource references

EJB references, that is logical names (or “nicknames”) for EJBs that were to the actual EJB JNDI names. Follow these steps to bind an EJB local reference to a JNDI name:

1. From the J2EE view, double-click the **WebbankEntityEJBs** module to open its deployment descriptor.
2. Switch to the References page.
3. Expand the tree under the Customer bean and select the **ejb/CustomerAccount** reference, as shown in Figure 14-14 on page 673.
4. In the WebSphere bindings section, specify `webbank/ejb/CustomerAccount`.
5. Repeat these steps for all references in this EJB module.
6. Save the deployment descriptor.
7. Repeat all steps for the WebbankSessionEJBs module and the WebbankWeb module, using Table 14-2 on page 673 as a guide for JNDI and resources names.



The following are binding properties for the WebSphere Application Server.

JNDI name: webbank/ejb/Customer

Figure 14-14 Setting EJB References bindings

Note: EJB references bindings can be defined/overridden at deployment time in the administrative console for all modules except for application clients, for which you must use the Assembly Toolkit.

Table 14-2 EJB and resource references - JNDI names list

EJB/Resource Reference	Corresponding JNDI Name
ejb/BranchAccount	webbank/ejb/BranchAccount
ejb/CustomerAccount	webbank/ejb/CustomerAccount
ejb/Customer	webbank/ejb/Customer
ejb/TransferStateless	webbank/ejb/TransferStateless
ejb/TransferStateful	webbank/ejb/TransferStateful
ejb/Consultation	webbank/ejb/Consultation
jms/webbankQCF	webbank/jms/webbankQCF
jms/webbankQ	webbank/jms/webbankQ

14.2.4 Binding Web modules to virtual hosts

Web modules need to be bound to a specific virtual host. For our sample, we want to attach the Webbank application to the virtual host called webbank_vhost created in 14.1.5, "Creating the virtual host for IBM HTTP Server (or Apache)" on page 655. By associating a Web module to a specific virtual host, you tell the Web server plug-in that all requests that match this virtual host, such as `http://www.webbank.itso.ibm.com:90/webbank`, must be handled by the Webbank Web application.

You must bind all Web modules to a virtual host. To do this:

1. In the J2EE view, double-click the **webbankWeb** Web module to edit its deployment descriptor.
2. On the Overview page, find the WebSphere bindings section.
3. Supply `webbank_vhost` as the virtual host name.
4. Save the deployment descriptor file.

14.2.5 Binding the message-driven bean to a JMS listener

The TransferMDB message-driven bean must be configured to listen to a specific queue or topic. You must now link this message-driven bean to the JMS listener created in "Creating the JMS listener port" on page 667.

To do this using the Assembly Toolkit, follow these steps:

1. In the J2EE view, double-click the **WebbankSessionEJB** module to edit its deployment descriptor.
2. Switch to the Beans page and select the **TransferMDB** bean.
3. In the WebSphere bindings section, set the ListenerPortName to `WebbankMDBListener`.
4. Save the deployment descriptor file.

All bindings are now complete. You should save the ready-to-deploy EAR file.

14.3 Generating deployment code

At some point, you will need to generate the deployment code for the Enterprise JavaBeans. You can do this in the WebSphere Studio application development tool, in the Assembly Toolkit, from the command line, or at deployment time. In general, we recommend that you regenerate the code with the EJBDeploy tool corresponding to the runtime. For example, generate the code with the Assembly Toolkit if you deploy in WebSphere Application Server 5.1, or with the AAT if you deploy to WebSphere Application Server 5.0.2.

14.3.1 Using the Assembly Toolkit

To generate the EJB deployment code from the Assembly Toolkit, follow these steps:

1. Select the EJB module for which you want to generate the code.
1. Select **Generate -> Deployment and RMIC code...**.
2. Select all the EJBs in the list.
3. Click **Finish**.

14.3.2 Using EJBDeploy

You can also generate the EJB deployed code using the EJBDeploy command-line tool. The EJBDeploy tool syntax is shown in Example 14-2.

Example 14-2 EJB deploy syntax

```
EJBDeploy (v5.0, 20031007_1915-WB212-AD-V511D-W5)

Syntax: EJBDeploy inputEar workingDirectory outputEar [options]
Options:
  -cp "jar1;jar2"   List of jar filenames required on classpath
  -codegen          Only generate the deployment code, do not run RMIC or Javac
  -bindear:options  Bind references within the EAR
  -dbschema schema  The name of the schema to create
  -dbvendor DBTYPE  Set the database vendor type, to one of:
         DB2UDB_V72      DB2UDB_V81      CLOUDSCAPE_V5
         DB2UDBOS390_V6  DB2UDBOS390_V7  DB2UDBISERIES
         INFORMIX_V73    INFORMIX_V93
         MSSQLSERVER_V7  MSSQLSERVER_2000
         ORACLE_V8       ORACLE_V9I
         SYBASE_V1200    SYBASE_V1250
  -debug            Compile the code with java debug information
  -keep             Do not delete the contents of the working directory
  -ignoreErrors     Do not halt for compilation or validation errors
  -quiet            Only display errors, suppress informational messages
```

```
 -nowarn          Disable warning and informational messages
 -noinform        Disable informational messages
 -rmic "options"  Set additional options to use for RMIC
 -35              Use the WebSphere 3.5 top-down mapping rules
 -40              Use the WebSphere 4.0 top-down mapping rules
 -trace           Trace progress of the deploy tool
 -sqlj            Use SQLJ instead of JDBC.
```

Example 14-3 shows a sample EJBDeploy run using the Webbank EJB module.

Example 14-3 EJBDeploy sample run

```
E:\WebSphereHandbooks\V51HandbookSource\Code>f:\WebSphere\appserver51\bin\ejbde
p
loy.bat WebbankV51-0129.ear f:\temp DeployedWebbank.ear -cp
.\WebbankV51\webbank
Common.jar -dbvendor DB2UDB_V81
Starting workbench.
Creating the project.
Building: /webbankSessionEJBs.
Building: /webbankEntityEJBs.
Deploying jar webbankSessionEJBs
Validating
[*Warning] ejbModule/META-INF/ejb-jar.xml(webbankSessionEJBs): CHKJ2875E:
<ejb-client-jar> webbankSessionEJBsClient.jar must exist in every EAR file that
contains this EJB module (EJB 1.1: 22.5, 23.4, 23.6).
Generating deployment code
Refreshing: /webbankSessionEJBs/ejbModule.
Building: /webbankSessionEJBs.
Invoking RMIC.
Generating DDL
Deploying jar webbankEntityEJBs
Validating
[*Warning] ejbModule/META-INF/ejb-jar.xml(webbankEntityEJBs): CHKJ2875E:
<ejb-client-jar> webbankEntityEJBsClient.jar must exist in every EAR file that
contains this EJB module (EJB 1.1: 22.5, 23.4, 23.6).
Generating deployment code
Refreshing: /webbankEntityEJBs/ejbModule.
Building: /webbankEntityEJBs.
Invoking RMIC.
Generating DDL
Generating DDL
Generating DDL
Writing output file
Shutting down workbench.
EJBDeploy complete.
0 Errors, 2 Warnings, 0 Informational
Messages
```

Tip: WebSphere Application Server V5 also provides a set of Ant tasks that you can use to automate the packaging/deployment of your applications. One of those tasks allows you to call EJBDeploy. Search for “Ant tasks” in the InfoCenter for more details.

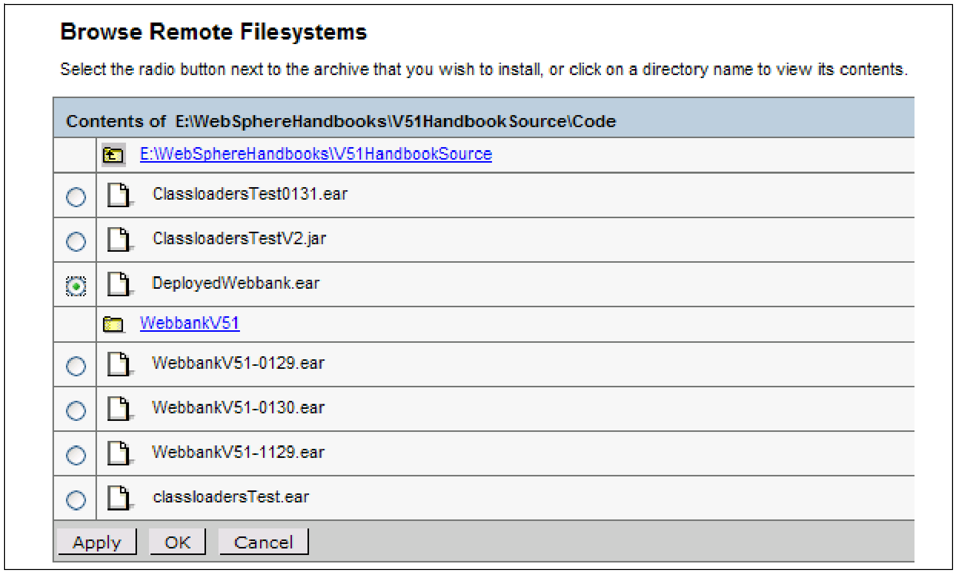
14.4 Deploying the application

We now have an enterprise application file that is ready to be deployed.

Note: If you have used the Assembly Tool to modify the application deployment settings and bindings, you have to export the complete EAR file by selecting **File -> Export...**, selecting the EAR file, and following the wizard instructions.

Additionally, we have created all the resources this application requires. You can now follow these steps to deploy the application:

1. Select **Applications -> Install New Application** from the administrative console navigation bar.
2. Click **Install**. Installable EAR files can be found locally or remotely (on any node in the current cell), as shown in Figure 14-15. Locate the deployed version of the Webbank application EAR file, select it, and click **OK**.


Browse Remote Filesystems

Select the radio button next to the archive that you wish to install, or click on a directory name to view its contents.

Contents of E:\WebSphereHandbooks\V51HandbookSource\Code

E:\WebSphereHandbooks\V51HandbookSource

( ) ClassloadersTest0131.ear

( ) ClassloadersTestV2.jar

(●) DeployedWebbank.ear

WebbankV51

( ) WebbankV51-0129.ear

( ) WebbankV51-0130.ear

( ) WebbankV51-1129.ear

( ) classloadersTest.ear

Figure 14-15 Browsing remote file systems

3. Click **Next**.
4. The next window lets you specify default bindings for the application you are deploying. Unless you check the "Override" option, bindings already specified in the EAR are not altered. The various bindings you can specify in this page are documented in Table 14-3. If you do choose to override bindings, make sure to select **Generate Default Bindings** at the top of this window to apply changes to the application you are deploying.

   In this example, the bindings were set in the application EAR file using the Assembly Toolkit. There is no need to override them. The defaults are also correct. Click **Next**.

Table 14-3 Application default bindings

Binding Name	Detailed Information
EJB prefix	You may generate default EJB JNDI names using a common prefix. EJBs for which you did not specify a JNDI name will get a default one built by concatenating the prefix and the EJB name. If you specify a prefix of webbank/ejb, then JNDI names default to webbank/ejb/EJBName, such as webbank/ejb/Consultation.
Override	Whether you want to override the current bindings. By default, existing bindings are not altered.
EJB 1.1 CMP bindings	Lets you bind all EJB 1.1 CMP entity beans to a specific data source, including user ID and password.
Connection Factory bindings	Lets you bind all EJB modules to a specific data source. You will have to go to the next window to override this setting at the EJB level.
Virtual host bindings	Lets you bind all Web modules to a specific virtual host, such as webbank_vhost.
Specify bindings file	You may also create a specific bindings file using your favorite editor and load it during application installation by clicking **Browse** next to the specific bindings file. For information on using a bindings file, see 14.4.1, "Using a bindings file" on page 681.

   The rest of the wizard is divided into steps.

5. Step 1: Provide options to perform the installation

   Step 1 gives you a chance to review the installation options. You can specify various deployment options such as JSP precompiling, whether you want to generate EJB deployment code, or enable reloadable classloaders for Web modules. In this example the defaults are correct.

   In our example we will take the defaults. Click **Next**.

Note: Steps 2 - 9 allow you to define bindings. We have already taken care of this using the Assembly Toolkit. You can skip directly to Step 10 if you like.

6. Step 2: Choose the current back-end ID

   Since version 5.0.2, it has been possible to select which back end the application entity EJBs will be targeting during deployment. A single EAR file can contain multiple database mappings. At deployment time, you can choose which one you want to use. In this case, this should be set to `UDB V8.1` since this is the version we are targeting at this time.

7. Step 3: Provide listener ports for messaging beans

   In 14.2.5, "Binding the message-driven bean to a JMS listener" on page 674, we used the Assembly Toolkit to set the listener port to `WebbankMDBListener`. You should see this value in the window.

   Click **Next**.

8. Step 4: Provide JNDI names for beans

   Use this window to bind the enterprise beans in your application or module to a JNDI name. In 14.2.1, "Defining EJB JNDI names" on page 669 we set these values using the Assembly Toolkit. You should see these values in this window. Click **Next**.

9. Step 5: Provide default data source mapping for modules containing 2.0 entity beans

   Specify the default data source for the EJB 2.x module containing 2.x CMP beans. In 14.2.2, "Defining data sources for entity beans" on page 671 we defined the JNDI name for webbankEntityEJBs to webbank/jdbc/webbank. You should see this in the window.

   Click **Next**.

10. Step 6: Map data sources for all 2.0 CMP beans

    Specify an optional data source for each 2.x CMP bean. Mapping a specific data source to a CMP bean will override the default data source for the module containing the enterprise bean.

    We do not need to do anything here. Click **Next**.

11. Step 7: Map EJB references to beans

    Each EJB reference defined in your application must be mapped to an enterprise bean. We used the Assembly Toolkit to do this in 14.2.3, "Binding EJB and resource references" on page 672.

    Click **Next**.

12. Step 8: Map resource references to resources

    Each resource reference defined in the application must be mapped to the corresponding resource. We have already mapped the JMS resources used by the Web module in the Assembly Toolkit in 14.2.3, “Binding EJB and resource references” on page 672.

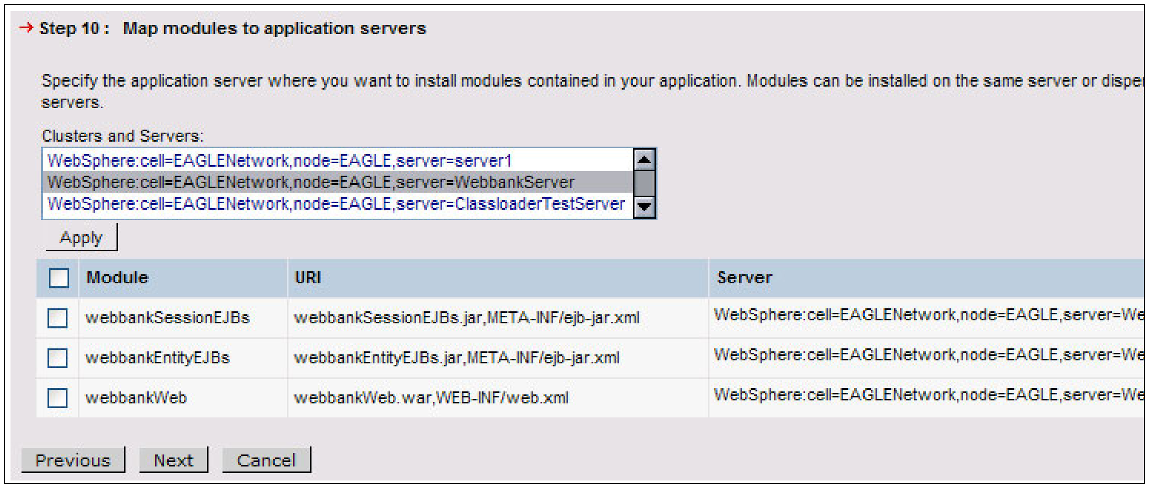
13. Step 9: Map virtual hosts for Web modules

    For each Web module, select the virtual host we created for the application (webbank_vhost).

14. Step 10: Map modules to application servers

    Select the server that each module should be deployed on. For better performance, we recommend that you deploy all modules from one application in a single server.

    Select the modules from the Webbank application, select **WebbankServer**, and click **Apply**.



→ Step 10 : Map modules to application servers

Specify the application server where you want to install modules contained in your application. Modules can be installed on the same server or dispe servers.

Clusters and Servers:

WebSphere:cell=EAGLENetwork,node=EAGLE,server=server1
WebSphere:cell=EAGLENetwork,node=EAGLE,server=WebbankServer
WebSphere:cell=EAGLENetwork,node=EAGLE,server=ClassloaderTestServer

☐	Module	URI	Server
☐	webbankSessionEJBs	webbankSessionEJBs.jar,META-INF/ejb-jar.xml	WebSphere:cell=EAGLENetwork,node=EAGLE,server=We
☐	webbankEntityEJBs	webbankEntityEJBs.jar,META-INF/ejb-jar.xml	WebSphere:cell=EAGLENetwork,node=EAGLE,server=We
☐	webbankWeb	webbankWeb.war,WEB-INF/web.xml	WebSphere:cell=EAGLENetwork,node=EAGLE,server=We

Figure 14-16 Mapping modules to application servers

    Click **Next**.

15. Step 11: Ensure all unprotected 2.0 methods have the correct level of protection

    By default, EJB methods are unprotected. On this window, you can elect to refuse all calls to unprotected methods, or specify which methods you want to exclude.

    Since we are not addressing security in this book, leave the settings as is and click **Next**.

16. Step 12: Summary

    The Summary window gives an overview of application deployment settings. If those settings are fine, click **Finish** to deploy the application.

17. Save the configuration.

Deployment is now complete. Since we have added a new application to the configuration, you should now regenerate the Web server plug-in configuration and move the new configuration to the Web server. For a quick refresh, restart the Web server.

You can now start the WebbankServer application server and try the application by invoking `http://<webbank_vhost_alias>/webbank`. Make sure that the JMS server has been started, and that the host name is one of the three names you have added to the webbank_vhost definition (see 14.1.5, “Creating the virtual host for IBM HTTP Server (or Apache)” on page 655).

14.4.1 Using a bindings file

If you want to use a bindings file when you install an application, you can load it by clicking **Browse** next to the Specific bindings file option. When using this file you should only specify bindings that differ from the defaults, not the full bindings strategy. For example, if you wanted to bind all EJBs except one to the webbank data source, you would create an XML file like this:

```
<?xml version="1.0"?>
<!DOCTYPE dfltbndngs SYSTEM "dfltbndngs.dtd">
<!-- This DTD is located under <WAS_HOME>\bin\wsinstance\propdefaults -->
<dfltbndngs>
   <module-bindings>
      <ejb-jar-binding>
         <jar-name>webbankEntityEJBs.jar</jar-name>
         <!-- This jar-name must correspond to the name provided in the
application.xml file -->
         <ejb-bindings>
            <ejb-binding>
               <ejb-name>Customer</ejb-name>
               <!-- for test only: Overriding module level datasource -->
               <connection-factory>
                  <jndi-name>webbank/jdbc/webbank2</jndi-name>
                  <res-auth>PerConnFact</res-auth>
               </connection-factory>
            </ejb-binding>
         </ejb-bindings>
      </ejb-jar-binding>
   </module-bindings>
</dfltbndngs>
```

For more information about creating specific bindings file, refer to the InfoCenter.

14.5 Deploying application clients

There are two types of application clients you can deploy:

- J2EE application client: This client is a Java application program that accesses EJBs, JDBC databases, and JMS message queues. The J2EE application client program runs on client machines. This program allows the same Java programming model as other Java programs; however, the J2EE application client depends on the application client runtime to configure its execution environment, and it uses the JNDI name space to access resources.

  The J2EE application client brings the J2EE programming model to the client, and provides:

  - XML deployment descriptors
  - J2EE naming (java:comp/env) including EJB references and resource references
- Java thin application client: This client provides a lightweight Java client programming model. This client is best suited for use in situations where a Java client application exists but the application must be enhanced to make use of EJBs, or where the client application requires a thinner, more lightweight environment than the one offered by the J2EE application client.

You can use the WebSphere J2EE client and thin client installation programs to install the necessary WebSphere runtime on a client machine, as well as the correct JRE. The J2EE client also comes with the J2EE client container. This J2EE client container is also installed as part of a full WebSphere install. In other words, if you already have installed WebSphere, you do not need to install the WebSphere J2EE client on top. The client install programs can be found on a separate CD in the WebSphere box.

14.5.1 Defining application client bindings

We have two J2EE clients: one for consulting accounts, the other one to transfer money between accounts. The first thing you need to do is to configure the bindings for these. This can only be done in the AAT. For those specific client applications, only EJB references need to be defined.

You need to specify the complete naming structure to reach the server where the EJBs are deployed. For example, the machine where we deployed the application for testing has the Network Deployment version installed. EAGLE is the name of the node where the WebbankServer resides.

Therefore, the ejb/Consultation EJB reference can be bound to:

```
cell/nodes/EAGLE/servers/WebbankServer/webbank/ejb/Consultation
```

Similarly, the ejb/Transfer EJB reference can be bound to:

```
cell/nodes/EAGLE/servers/WebbankServer/webbank/ejb/TransferStateful
```

If you have created a cluster of application servers, you should use:

```
cell/clusters/<clusterName>/webbank/ejb/Consultation
```

You will also need to change the provider URL according to the target server.

Tip: The **`dumpNameSpace`** command can help you better understand the naming structure, since results show the different entries (logical names) in the naming tree and what they correspond to. For example, on the EAGLE system, a portion of the dumpNameSpace results look like this:

```
============================================================================
Name Space Dump
   Provider URL: corbaloc:iiop:localhost:2809
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: cell
   Starting context: (top)=EAGLENetwork
   Formatting rules: jndi
   Time of dump: Sun Feb 15 21:55:47 CET 2004
============================================================================
Beginning of Name Space Dump
============================================================================
    1 (top)
    2 (top)/cellname                                    java.lang.String
    3 (top)/legacyRoot
10 (top)/clusters                                      javax.naming.Context
   11 (top)/clusters/WebbankCluster
javax.naming.Context
   11    Linked to URL:
corbaloc::EAGLE.mas.es.ibm.com:2810,:EAGLE.mas.es.ibm.com:2811/NameServiceSe
rverRoot
23 (top)/nodes/EAGLE/servers/ClassloaderTestServer    javax.naming.Context
   23    Linked to URL:
corbaloc::EAGLE.mas.es.ibm.com:9811/NameServiceServerRoot
```

14.5.2 Exporting the EAR to the client system

You then have to copy the EAR file to the client machine. Although you do not need the complete contents of the EAR file to run the application client (for example the Web modules), it is better to keep a single EAR file, mainly for maintenance purposes.

14.5.3 Launching the J2EE client

A J2EE client needs a container to run in. This container can be started using the launchClient program, which can be found under <WAS_HOME>/bin. The launchClient program has the following syntax:

```
Usage: launchClient [<userapp.ear> | -help | -?] [-CC<name>=<value>] [app args]
```

Where:

`<userapp.ear>`	The path/name of the .ear file containing the client application.
`-help, -?`	Print this help message.

The -CC properties are for use by the Application Client Runtime:

`-CCverbose`	<true | false> Use this option to display additional informational messages. The default is false.
`-CCclasspath`	A classpath value. When an application is launched, the system classpath is not used. If you need to access classes that are not in the EAR file or part of the resource classpaths, specify the appropriate classpath here. Multiple paths may be concatenated.
`-CCjar`	The name of the client JAR file within the EAR file that contains the application you wish to launch. This argument is only necessary when you have multiple client JAR files in the EAR file.
`-CCBootstrapHost`	The name of the host server you wish to connect to initially. The format is `your.server.ofchoice.com`.
`-CCBootstrapPort`	The server port number. If not specified, the WebSphere default value (2809) is used.
`-CCproviderURL`	Provides bootstrap server information that the initial context factory can use to obtain an initial context. WebSphere Application Server initial context factory can use either a CORBA object URL or an IIOP URL. CORBA object URLs are more flexible than IIOP URLs and are the recommended URL format to use. This value can contain

	more than one bootstrap server address. This feature can be used when attempting to obtain an initial context from a server cluster. In the URL, you can specify bootstrap server addresses for all servers in the cluster. The operation will succeed if at least one of the servers is running, eliminating a single point of failure. The address list does not process in a particular order. For naming operations, this value overrides the `-CCBootstrapHost` and `-CCBootstrapPort` parameters. An example of a CORBA object URL specifying multiple systems is: `-CCproviderURL=corbaloc:iiop:myserver.mycompany.com :9810,:mybackupserver.mycompany.com:2809`
`-CCinitonly`	<true | false> This option is intended for ActiveX applications to initialize the Application Client runtime without launching the client application. The default is false.
`-CCtrace`	<true | false> Use this option to have WebSphere write debug trace information to a file. You may need this information when reporting a problem to IBM Service. The default is false.
`-CCtracefile`	The name of the file to write trace information. The default is to output to the console.
`-CCpropfile`	Name of a properties file containing launchClient properties. In the file, specify the properties without the `-CC` prefix. For example: `verbose=true`.

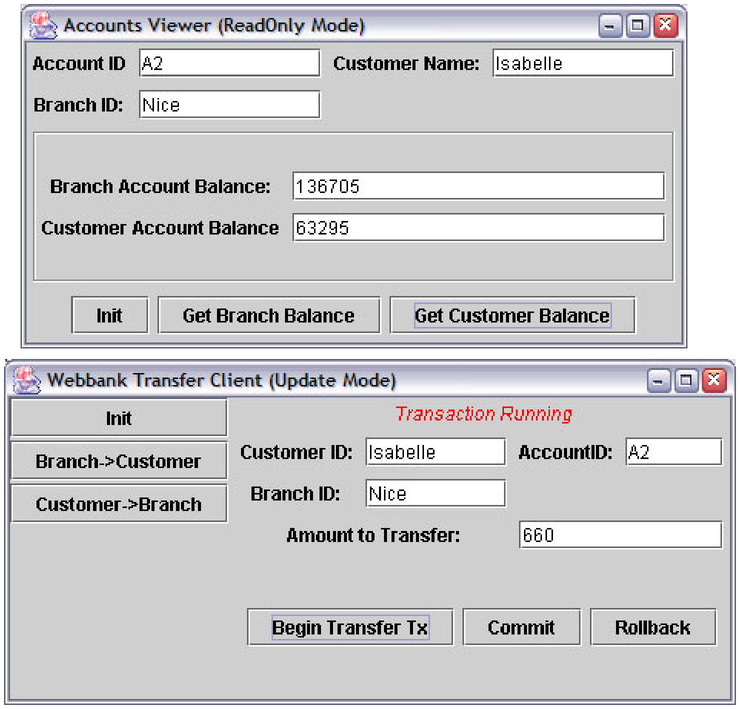
The `app args` are for use by the client application and are ignored by WebSphere.

You can start the Webbank application clients, shown in Figure 14-17, as follows:

Example 14-4 Launching an application client

```
F:\WebSphere\AppServer50\bin>launchClient.bat C:\WebbankFiles\webbankv51.ear
-CCjar=webbankTransferClient.jar -CCproviderURL=corbaloc:iiop:myserv:9811
IBM WebSphere Application Server, Release 5.1
J2EE Application Client Tool
Copyright IBM Corp., 1997-2003
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application Client Environment.
WSCL0035I: Initialization of the J2EE Application Client Environment has
completed.
WSCL0014I: Invoking the Application Client class
itso.webbank.testclients.transfer.Trans
tClient
```

Since we are providing two J2EE clients, you must use the `-CCjar` option to specify which client to launch.



Account ID A2 Customer Name: Isabelle
Branch ID: Nice
Branch Account Balance: 136705
Customer Account Balance 63295

Customer ID: Isabelle AccountID: A2
Branch ID: Nice
Amount to Transfer: 660

Figure 14-17 Webbank J2EE clients

You need first to supply the account information, then click **Init**. Then, you can get the branch and customer account balances or make transfers.

14.6 Understanding WebSphere classloaders

The classloaders used by WebSphere have been much changed in WebSphere Application Server V5. This section starts by giving a little background on Java 2 classloaders and how they work. Then, we describe the different WebSphere classloaders. Finally, we explain how you can customize the behavior of WebSphere classloaders.

14.6.1 A brief introduction to Java 2 classloaders

Classloaders enable the Java virtual machine (JVM) to load classes. Given the name of a class, the classloader should locate the definition of this class. Each Java class must be loaded by a classloader. The bootstrap classloader is responsible for loading the core Java libraries, that is <JAVA_HOME>/lib/rt.jar and <JAVA_HOME>/lib/i18n.jar. This classloader, which is part of the core JVM, is written in native code.

When you start a JVM, you use three classloaders: the bootstrap classloader mentioned previously, the extensions classloader, and the system classloader.

- Extensions classloader: The extensions classloader is responsible for loading the code in the extensions directories (<JAVA_HOME>/lib/ext or any other directory specified by the `java.ext.dirs` system property). This classloader is implemented by the sun.misc.Launcher$ExtClassLoader class.
- System classloader: The system classloader is responsible for loading the code that is found on java.class.path, which ultimately maps to the system CLASSPATH variable. This classloader is implemented by the sun.misc.Launcher$AppClassLoader class.

Delegation is a key concept to understand when dealing with classloaders. It states that a custom classloader delegates class loading to its parent before trying to load the class itself. The parent classloader can either be another custom classloader or the bootstrap classloader. Another way to look at this is that a class loaded by a specific classloader can only reference classes that this classloader or its parents can load (but not its children).

The extensions classloader is the parent for the system classloader. The bootstrap classloader is the parent for the extensions classloader. The classloaders hierarchy is shown in Figure 14-18 on page 688.

If the system classloader needs to load a class, it first delegates to the extensions classloader, which in turn delegates to the bootstrap classloader. If the parent classloader cannot load the class, the child classloader tries to find the class in its own repository. In this manner, a classloader is only responsible for loading classes that its ancestors cannot load.

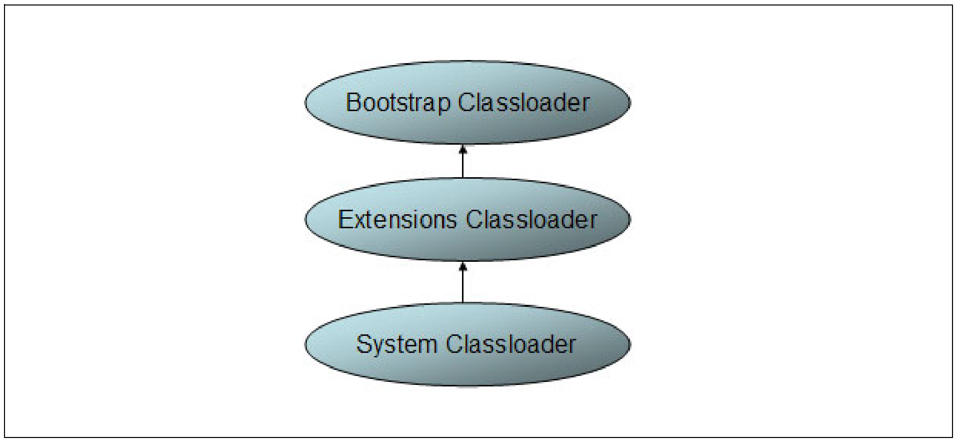


Figure 14-18 Java classloaders hierarchy

This behavior can lead to some interesting problems if a class is loaded from a classloader that is not on a leaf node in the classloader tree. Consider Example 14-5: a class called WhichClassLoader1 loads a class called WhichClassLoader2, which in turn invokes a class called WhichClassLoader3.

Example 14-5 WhichClassLoader1 and WhichClassLoader2 source code

```
public class WhichClassLoader1 {

   public static void main(String[] args)
      throws javax.naming.NamingException
   {
      // Getting Classpathes Value
      StringBuffer bootstrapClassPath =
         new StringBuffer(System.getProperty("sun.boot.class.path"));
      StringBuffer extClassPath =
         new StringBuffer(System.getProperty("java.ext.dirs"));
      StringBuffer systemClassPath =
         new StringBuffer(System.getProperty("java.class.path"));
      // Printing them out
      System.out.println("Bootstrap classpath=\t" +
                                       bootstrapClassPath + "\t" );
      System.out.println("\nExtension classpath=\t" + extClassPath + "\t");
      System.out.println("\nSystem classpath=\t" + systemClassPath + "\t\n");
      //Loading Classes
      Object obj = new Object();
      WhichClassLoader1 wcl1 = new WhichClassLoader1();
      WhichClassLoader2 wcl2 = new WhichClassLoader2();
      //Who loaded what?
      System.out.println("->Object was loaded by " +
                                obj.getClass().getClassLoader());
      System.out.println(
         "->WCL1 class was loaded by " + wcl1.getClass().getClassLoader());
      System.out.println(
         "->WCL2 class was loaded by " + wcl2.getClass().getClassLoader());
      wcl2.getTheClass();
   }
}
==========================================================================
public class WhichClassLoader2 {

   //This method is invoked from WhichClassLoader1.
   public void getTheClass() {
      WhichClassLoader3 wcl3 = new WhichClassLoader3();
      System.out.println("->WCL 3 was loaded
by:"+wcl3.getClass().getClassLoader());
   }
}
```

If all WhichClassLoaderX classes are put in the system classpath, the three classes are loaded by the system classloader, and this sample runs just fine. Now suppose you package the WhichClassLoader2.class file in a JAR file that you store under <JAVA_HOME>/lib/ext directory; you would then see the output listed in Example 14-6.

Example 14-6 NoClassDefFoundError exception trace

```
Bootstrap classpath=
F:\WSADV5\eclipse\jre\lib\rt.jar;F:\WSADV5\eclipse\jre\lib\i18n.jar;F:\WSADV5\e
clipse\jre\classes
Extension classpath=F:\WSADV5\eclipse\jre\lib\ext
System classpath=E:\WSADworkspaces\WebbankV5\ClassloadersTest\bin

->Object was loaded by null
->WCL1 class was loaded by sun.misc.Launcher$AppClassLoader@5059e39d
->WCL2 class was loaded by sun.misc.Launcher$ExtClassLoader@505ca39d
java.lang.NoClassDefFoundError:
com/ibm/wss/lge/classloaders/test/WhichClassLoader3
at
com.ibm.wss.lge.classloaders.test.WhichClassLoader2.getTheClass(WhichClassLoade
r2.java:6)
```

```
at
com.ibm.wss.lge.classloaders.test.WhichClassLoader1.main(WhichClassLoader1.java
:27)
Exception in thread "main"
```

As you can see, the program fails with a NoClassDefFoundError exception, which may sound strange since WhichClassLoader3 is on the system classpath. The problem is that it is on the wrong classpath.

As you can see from the trace, the WhichClassLoader2 class was loaded by the extensions classloader. In fact, the system classloader delegated the load of the WhichClassLoader2 class to the extensions classloader, which in turn delegated the load to the bootstrap classloader. Since the bootstrap classloader could not find the class, the class loading control got returned to the extensions classloader. The extensions classloader found the class, and therefore loaded it. Now, the extensions classloader needs to load the WhichClassLoader3 class. It delegates to the bootstrap classpath, which can't find the class, then tries to load it itself and does not find it either. A NoClassDefFoundError exception is thrown. Once a class is loaded by a classloader, any new classes that it tries to load will reuse the same classloader, or go up the hierarchy to find a class. *A classloader can only find classes up in the hierarchy, not down.*

14.6.2 WebSphere classloaders overview

WebSphere provides several custom delegated classloaders: the WebSphere extensions classloader and different application classloaders. The classloaders hierarchy is depicted in Figure 14-19 on page 691.

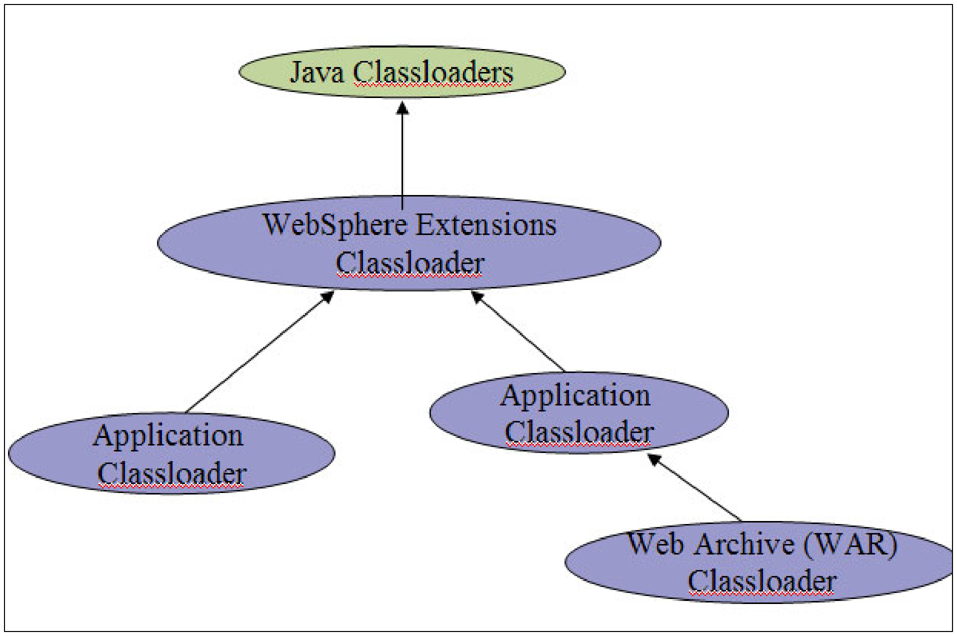


Figure 14-19 WebSphere classloaders hierarchy

WebSphere extensions classloader

The WebSphere extensions classloader, whose parent is the Java system classloader, is primarily responsible for the loading of the WebSphere class libraries in the following directories:

- <JAVA_HOME>\lib
- <WAS_HOME>\classes (Runtime Class Patches directory, or RCP)
- <WAS_HOME>\lib (Runtime classpath directory, or RP)
- <WAS_HOME>\lib\ext (Runtime Extensions directory, or RE)

The WebSphere runtime is loaded by the WebSphere extensions classloader based on the ws.ext.dirs system property, which is initially derived from the WS_EXT_DIRS environment variable set in the setupCmdLine script file. The default value of ws.ext.dirs is the following:

```
SET
WAS_EXT_DIRS=%JAVA_HOME%/lib;%WAS_HOME%/classes;%WAS_HOME%/lib;%WAS_HOME%/
lib/ext;%WAS_HOME%/web/help;%ITP_LOC%/plugins/com.ibm.etools.ejbdeploy/
runtime
```

The RCP directory is intended to be used for fixes and other APARs that are applied to the application server runtime. These patches override any copies of the same files lower in the RP and RE directories. The RP directory contains the core application server runtime files. The bootstrap classloader first finds classes in the RCP directory then in the RP directory. The RE directory is used for extensions to the core application server runtime.

Each directory listed in the ws.ext.dirs environment variable is added to the WebSphere extensions classloaders classpath. In addition, every JAR file and/or ZIP file in the directory is added to the classpath.

You can extend the list of directories/files loaded by the WebSphere extensions classloaders by setting a ws.ext.dirs custom property to the Java virtual machine settings of an application server.

Important: One of the main changes from WebSphere Application Server V4.0 is that the J2EE runtime is loaded by the Java system classloader (which is the parent of the WebSphere extensions classloader). This lets you put classes that depend on the J2EE runtime, such as database drivers, on the application server classpath (as opposed to listing them on ws.ext.dirs as you had to do in Version 4.0). However, a preferred method to load such classes is to use the new shared libraries support provided in Version 5.0 (see “Shared libraries” on page 695).

14.6.3 Application and Web module classloaders

Applications consist of five primary elements: Web modules, EJB modules, application client modules, resource adapters (RAR files), and dependency JARs. Dependency JAR files are also known as utility JAR files, that is JAR files containing code used by both EJBs and/or servlets. An XML parser or some utility framework are good examples of utility JAR files.

EJB modules, dependency JARs, resource adapters files, and shared libraries associated to that application are always grouped together into the same classloader. This classloader is called the application classloader. Depending on the application classloader policy, this application classloader can be shared by multiple applications (EAR) or can be unique for each application.

By default, Web modules receive their own classloader (a WAR classloader) to load the contents of the WEB-INF/classes and WEB-INF/lib directory. The default behavior can be modified by changing the application's WAR classloader policy (the default being Module). If the WAR classloader policy is set to `Application`, the Web module contents are loaded by the *application classloader* (in addition to the EJBs, RARs, dependency JARs, and shared libraries). The application classloader is the parent of the WAR classloader.

14.6.4 Understanding classloader policies

For each application server in the system, the application classloader policy can be set to `Single` or `Multiple`.

When the application classloading policy is set to `Single`, a single application classloader is used to load all EJBs, dependency JARs, and shared libraries within the application server (JVM). If the WAR classloader loading policy has been set to `Application`, the Web module contents for this particular application are also loaded by this single classloader.

When the application classloading policy is set to `Multiple`, each application will receive its own classloader for loading EJBs, dependency JARs, and shared libraries. Depending on whether the WAR classloader loading policy is set to `Module` or `Application`, the Web module may or may not receive its own classloader.

But let's take an example. Say you have two applications: Application1 and Application2. Each application has an EJB module and a Web module (App1EJBs.jar/App1Web.war and App2EJBs.jar/App2Web.war respectively). Each application uses dependency JARs (Dependency1.jar and Dependency2.jar).

Now, say you have done the following:

- Set the application classloader policy to be Single
- Set the App1Web module classloader policy to be Module
- Set the App2Web module classloader policy to be Application

You would obtain a classloader hierarchy (with their respective classpath) as depicted in Figure 14-20 on page 694. Note that the classpaths listed there are just examples; many parameters can affect what is actually put on the classloader's classpath.

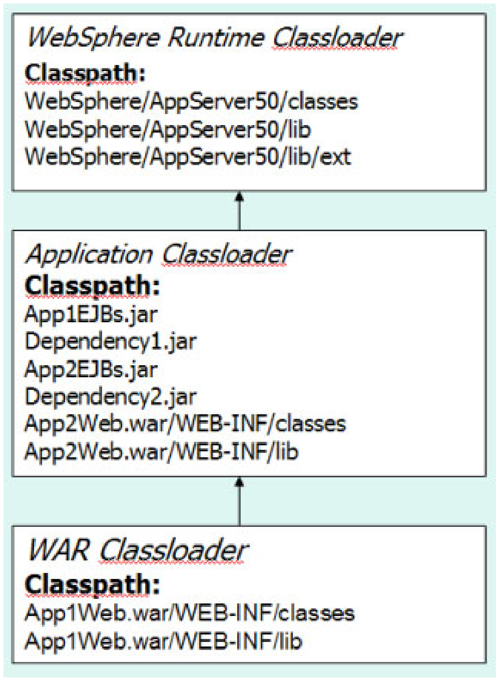


Figure 14-20 Application Classloader Policy: Single

Now, say you are changing the application classloader policy to Multiple but leave the Web module policies unchanged. You would then obtain something similar to Figure 14-21 on page 695.

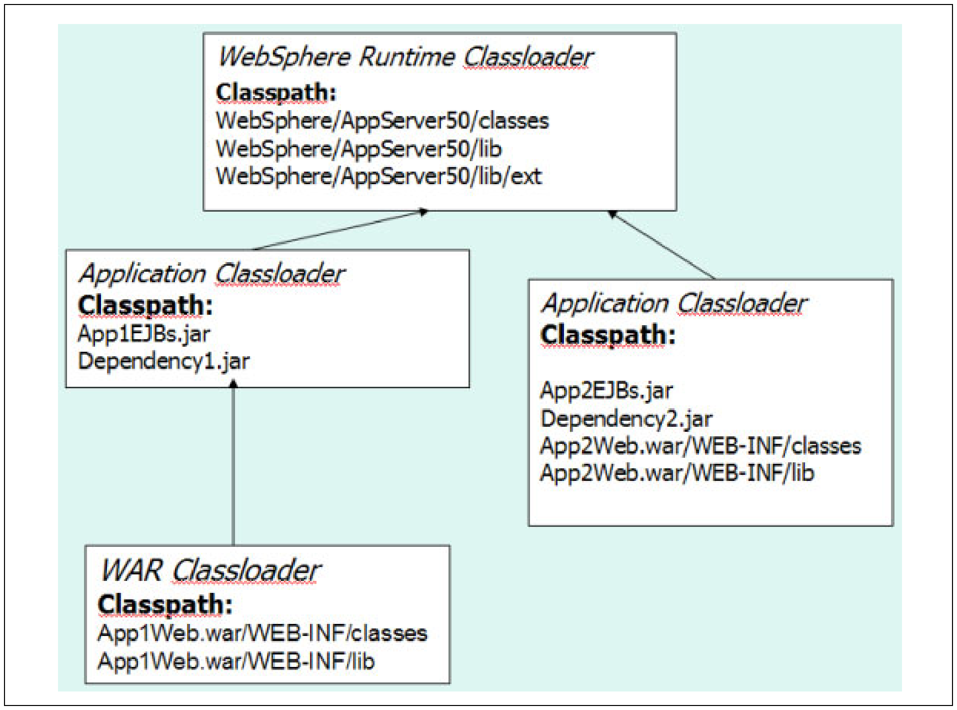


Figure 14-21 Application Classloading Policy: Multiple

14.6.5 Understanding classloading/delegation mode

There are two possible values for a classloader loading mode: PARENT_FIRST and PARENT_LAST. The default value for classloading mode is PARENT_FIRST. This policy causes the classloader to first delegate the loading of classes to its parent classloader before attempting to load the class from its local classpath. This is the default policy for standard JVM classloaders. If the classloading policy is set to `PARENT_LAST`, the classloader attempts to load classes from its local classpath before delegating the classloading to its parent. This policy allows an application classloader to override and provide its own version of a class that exists in the parent classloader.

Delegation mode can be set for the following classloaders: application classloader, WAR classloader, and shared library classloader.

14.6.6 Shared libraries

You would typically use shared libraries to point to a set of JARs (for example, a framework), and associate those JARs to an application or application server.

Shared libraries are especially useful when you have different versions of a same framework you want to associate to different applications.

Shared libraries consist of a symbolic name, a Java classpath, and a native path (for loading JNI libraries), and can be defined at the cell, node, or server level. However, defining a library at one of the three levels does not cause the library to be loaded. You must associate the library to an application and/or application server in order for the classes represented by the shared library to be loaded.

If you associate the classloader to an application, the JARs listed on the shared library path are loaded by the application classloader (together with EJB JARs, RARs and dependency JARs). If you associate the shared library at the application server level, the JARs listed on the shared library path are loaded by a specific classloader (which you have to define).

See “Step 4: Sharing dependency JARs among multiple applications” on page 702 for more details.

14.6.7 Application extensions classloader

In WebSphere Application Server Version 4.0, you had the possibility to drop JAR files under <WAS_HOME>/lib/app to share those JARS across all applications. This is still supported, but is not recommended for new deployment. It is mainly provided for compatibility with WebSphere V4.0 deployments. The main issue with this classloader is that its contents are seen by *all* applications. The recommended approach is to use shared libraries, which have a different behavior.

See “Step 4: Sharing dependency JARs among multiple applications” on page 702 for more details.

14.6.8 Reloadable classloaders

The application and the Web module classloaders are reloadable classloaders. They monitor changes in the application code to automatically reload modified classes. This behavior can be altered at deployment time.

14.7 Learning classloaders by example

We have now described all the different options for influencing classloader behavior. Let’s take an example and use all the different options we have discussed to this point so that you can better evaluate what is the best solution for your application(s).

We have created a very simple application, with one JSP and one EJB. Both call a class, VersionChecker, shown in Example 14-7. This class can print out which classloader was used to load the class. The VersionChecker class also has an internal value that can be printed to check which version of the class we are actually using. This will be used later to demonstrate the use of multiple versions of the same utility JAR.

Example 14-7 VersionChecker class source code

```
public class VersionChecker {

   static final public String classVersion = "1.0";

   public String printClassLoaderInfo() {
      return ("Class Version Checker was loaded by: " +
                this.getClass().getClassLoader());
   }

   public String printVersionInfo () {
      return ("Class version is: " + classVersion);
   }
}
```

Once installed, the application can be invoked via `http://localhost/testclassloaders`. This invokes a JSP (callVersionChecker.jsp) which returns the following sample information:

```
Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@2410f589
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestWeb.war\WEB-INF\classes;F:\WebSphere\AppServer51\installedApps\
EAGLENetwork\ClassloaderTest.ear\ClassloaderTestWeb.war\WEB-INF\lib\Classlo
adersTest.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Classload
erTest.ear\ClassloaderTestWeb.war; Delegation Mode: PARENT_FIRST
Class version is: 1.0
```

The ClassloadersTest.jar file contains the VersionChecker class file. For all the following tests, we have, unless otherwise noted, left the classloader policies and loading modes to their defaults. In other words, we have one classloader for the application and one for the WAR file. Both have their delegation modes set to `PARENT_FIRST`. All tests have been done with the Network Deployment edition (however, working with the base product would not matter). We assume the application has been deployed to an application server called ClassloaderTestServer.

Example 14-8 CallVersionChecker.jsp source

```
...
<TITLE>WAS V5.1 Classloaders Info</TITLE>
</HEAD>
<BODY>
<h2> Version Checker Information </h2>
<h3> Direct call from Servlet to Version Checker Class </h3>
<%
 com.ibm.wss.lge.classloaders.VersionChecker versionChecker
            = new com.ibm.wss.lge.classloaders.VersionChecker();
 out.println("<br/>" + versionChecker.printClassLoaderInfo());
 out.println("<br/>" + versionChecker.printVersionInfo());
%>
<h3> Call the VersionChecker via Access bean-> enterprise session bean </h3>
<%
 com.ibm.wss.lge.ejbs.ClassLoaderTesterAccessBean versionCheckerAB
            = new com.ibm.wss.lge.ejbs.ClassLoaderTesterAccessBean();
 out.println("<br />" + versionCheckerAB.callVersionChecker());
%>
</BODY>
...
```

During the tests, we worked directly on the file system by modifying the contents of the <WAS_HOME>\installedApps\EAGLENetwork\ClassloaderTest.ear directory, which is obviously not a wise thing to do in production! For more information, see "Maintenance best practices" on page 711.

14.7.1 Step 1: Simple WAR packaging

Let's start with the following assumption: our utility class is only used by a servlet (in the JSP, we have commented out the lines that invoke the EJB part of the application). We have placed the ClassloadersTest.jar file under the WEB-INF/lib directory of the WAR file.

Tip: You would typically put under **WEB-INF/lib** JAR files used by a **single** Web module, or a JAR file that *only* this web module should see.

When we run the application in such a configuration, we obtain the following result:

```
Version Checker Information
Direct call from Servlet to Version Checker Class

Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@2410f589
```

```
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestWeb.war\WEB-INF\classes;F:\WebSphere\AppServer51\installedApps\
EAGLENetwork\ClassloaderTest.ear\ClassloaderTestWeb.war\WEB-INF\lib\Classlo
adersTestV1.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Classlo
aderTest.ear\ClassloaderTestWeb.war;
Delegation Mode: PARENT_FIRST
Class version is: 1.0
```

There are a few things we can learn from this trace:

1. The type of the WAR classloader is `com.ibm.ws.classloader.CompoundClassLoader`.
2. It searches classes in the following order:

```
ClassloaderTestWeb.war\WEB-INF\classes
ClassloaderTestWeb.war\WEB-INF\lib\ClassloadersTest.jar
ClassloaderTestWeb.war
```

The WEB-INF/classes folder is usually used for servlets and property files, while the WEB-INF/lib is used for JARs (*not* ZIPs!). Note that the classloader classpath is built at application startup, which is why you see only the ClassloadersTest JAR listed there. If multiple JARs had been placed under this folder, you would see all of them listed on the local classpath. The root of the WAR file is the next place where you can put code or properties classes.

14.7.2 Step 2: Sharing the dependency JAR among multiple modules

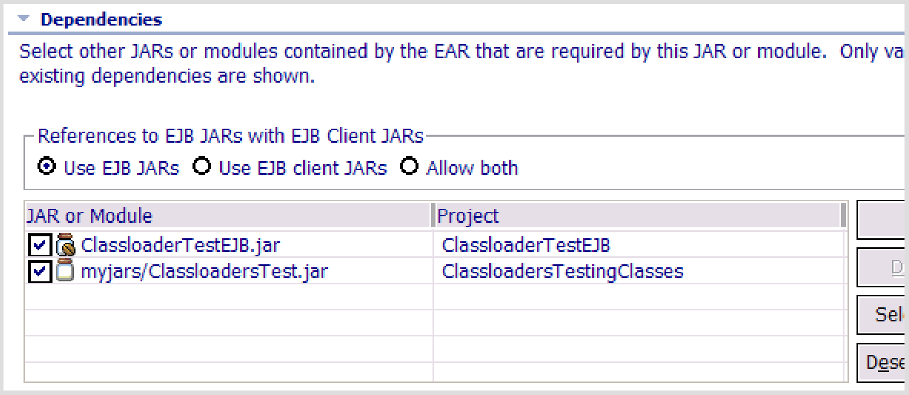
Next, we decide to run the EJB part of the application, which also depends on our ClassloadersTest.jar JAR file. Since the JAR file is to be used by multiple modules in the same application, the best solution is to place it relative to the root of the EAR file and to reference it via a Class-Path entry in the JAR manifest file.

The Manifest Class-Path entry

Let's say we place the ClassloadersTest.jar file in a myjars directory at the root of the EAR file. From a J2EE perspective, this directory is unknown, and the directory contents will not be added to the classpath automatically. The solution is to create a `Class-Path` entry in the META-INF/MANIFEST.MF file for each module which uses classes from the classloadersTest.jar file, as follows (paths are relative to the root of the EAR file):

```
#This is the Manifest file for the ClassloadersWeb.war file
Manifest-Version: 1.0
Class-Path: ClassloaderTestEJB.jar myjars/ClassloadersTest.jar
```

Tip: In you are using WebSphere Studio or the Assembly Toolkit to develop/package applications, we strongly advise that you use the MANIFEST.MF editor, shown below, to manage module dependencies.


Dependencies
Select other JARs or modules contained by the EAR that are required by this JAR or module. Only va
existing dependencies are shown.
References to EJB JARs with EJB Client JARs
Sel
Dese

[✓ Use EJB JARs] Use EJB client JARs Allow both

JAR or Module	Project
✓ ClassloaderTestEJB.jar	ClassloaderTestEJB
✓ myjars/ClassloadersTest.jar	ClassloadersTestingClasses

The test results are then as follows:

```
Version Checker Information
Direct call from Servlet to Version Checker Class

Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@2737a7e5 Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestEJB.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Cla
ssloaderTest.ear\myjars\ClassloadersTest.jar;
Delegation Mode: PARENT_FIRST
Class version is: 1.0

Call the VersionChecker via Access bean-> enterprise session bean
VersionChecker Classloader Info

Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@2737a7e5
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestEJB.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Cla
ssloaderTest.ear\myjars\ClassloadersTest.jar;
Delegation Mode: PARENT_FIRST
Class version is: 1.0
```

Although the classloader type is the same (CompoundClassLoader), this is actually the application classloader, *not* the WAR classloader (you can see this from the classpath, which is very different). Remember, all dependency JARs are loaded by the application classloader. As soon as you reference a JAR via a Class-Path directive, it is considered a dependency JAR.

As you can see, the *same* classloader is used by the servlet and the EJBs to load the VersionChecker class.

Tip: You should to place at the root of the EAR module JAR files which are common to the entire application, such as utility classes (the webbankCommon.jar of the Webbank application for example).

14.7.3 Step 3: Changing the WAR classloader delegation mode

What happened to the ClassloadersTest.jar under WEB-INF/lib? We did not say we had removed it! It is actually still there, but it is ignored since the delegation mode of the WAR is set to `PARENT_FIRST`. Now, let's set the delegation mode to `PARENT_LAST`. This can be done by following these steps:

1. Select the **Enterprise Applications** entry in the navigation area.
2. Select the **ClassloaderTest** application.
3. Scroll down until you see the **Web Modules** entry and select it.
4. Select the **ClassloaderTestWeb.war**.
5. Change the Classloader Mode to `PARENT_LAST`.
6. Click **OK**.
7. Save the configuration.
8. Restart the application.

Now, for the sake of demonstrating that a different version gets loaded, we will put the ClassloaderTestV2.jar under WEB-INF/lib. This version should return 2.0 for the class version. Here are the test results:

```
Direct call from Servlet to Version Checker Class
Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@610ee7e5
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestWeb.war\WEB-INF\classes;F:\WebSphere\AppServer51\installedApps\
EAGLENetwork\ClassloaderTest.ear\ClassloaderTestWeb.war\WEB-INF\lib\Classlo
adersTestV2.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Classlo
aderTest.ear\ClassloaderTestWeb.war;
Delegation Mode: PARENT_LAST
Class version is: 2.0
```

As expected, the code has now been loaded by the WAR classloader and the class version is 2.0.

Tip: Use this to specify that a Web module should use a specific version of a library such as Struts, or to override Xalan/Xerces versions coming with the WebSphere runtime. Put the common version at the top of the hierarchy, and the specialized version in WEB-INF/lib.

14.7.4 Step 4: Sharing dependency JARs among multiple applications

We are now in a situation where the ClassloaderTest.jar file is used by a single application. What if we wanted to share it among multiple applications? Of course, you could package it within each EAR file. But changes to this dependency JAR file would require redeploying all applications again. To avoid this, you can externalize “global” dependency JARs.

You basically have two solutions to this problem: one is to use the application extensions classloader, inherited from Version 4.0, and the other one (recommended) is to use shared libraries.

Using the application extensions classloader

The role of this classloader is to load any JAR file placed in the <WAS_HOME>/lib/app directory. This classloader has a delegation mode set to `PARENT_LAST`.

Note: All applications running within the same WebSphere node running with PARENT_FIRST delegation mode will “see” what is contained in this directory, regardless of the application server they run in. This might be what you want to achieve; you may have a framework that needs to be shared by all your applications. But if some applications want to use the level 1.0 of the framework while others want to use Version 2.0, using the lib/app directory is not the right solution.

By default, the <WAS_HOME>/lib/app directory does *not* exist. However, at application startup, the classloader runtime will check for its existence and bring up the application extensions classloader if it does exist.

Now, for testing purposes only, if you create this <WAS_HOME>/lib/app directory, drop the ClassloadersTestV2.jar in there, and restart the application server, here are the results you will obtain:

```
Direct call from Servlet to Version Checker Class
Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@272b6115
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestWeb.war\WEB-INF\classes;F:\WebSphere\AppServer51\installedApps\
EAGLENetwork\ClassloaderTest.ear\ClassloaderTestWeb.war\WEB-INF\lib\Classlo
adersTestV2.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Classlo
aderTest.ear\ClassloaderTestWeb.war;
Delegation Mode: PARENT_LAST
Class version is: 2.0

Call the VersionChecker via Access bean-> enterprise session bean
VersionChecker Classloader Info

Class Version Checker was loaded by:
com.ibm.ws.classloader.ExtJarClassLoader@2063425814
Local ClassPath: F:\WebSphere\AppServer51\lib\app\ClassloadersTestV2.jar;
Delegation Mode: PARENT_LAST
Class version is: 2.0
```

Note that the servlet still uses the code from WEB-INF/lib since its delegation mode is set to PARENT_LAST. The EJB, however, uses the code from the lib/app directory.

Using shared libraries

A much cleaner and manageable way to work is to use the new shared libraries support. Shared libraries can be defined at the cell/node/application server levels. Once you have defined a shared library, you must associate it to an application or to a server. JARs/folders listed on a shared library are always loaded by the application classloader.

You can define as many shared libraries as you wish. You can also associate multiple shared libraries with an application or application server.

Using Shared Libraries at the application level

Let’s define a shared library named VersionCheckerV2_SharedLib and associate it to our ClassloaderTest application.

This can be done as follows:

1. In the administrative console, go to **Environment-Shared Libraries**.

2. Select the level you want this shared library to be defined at, such as Cell, and click **New**.
3. Specify the following properties:
   - Name: `VersionCheckerV2_SharedLib`.
   - Description: Optional description of this library. However, it is always good to provide a description.
   - Classpath: The list of entries on the classpath. You must press Enter between each entry. Do not use any usual separators, such as ";" or ":". We highly recommend that if you need to provide an absolute path, you use WebSphere variables, such as %FRAMEWORK_JARS%/ClassloadersTestV2.jar. Make sure that you declare this variable at the same scope as the Shared Library (cell/node/server).
   - Native library path: A list of DLLs/.so files (used by JNI code).
4. Click **OK**.
5. Go to **Applications-Enterprise Applications**.
6. Select the **ClassloaderTest** application.
7. In Additional Properties, select **Libraries**.
8. Click **Add**. You should see the library you just created (if not, you probably have a scoping problem, so make sure you declared the Shared Variable at the right scope level).
9. Select the shared library, and click **OK**.
10. Save the configuration. Don't forget to synchronize the changes with the nodes if you are running the Network Deployment edition.

If we now remove the ClassloadersTest.jar file from the lib/app folder *and* restart the application server, we see the following test results:

```
Call the VersionChecker via Access bean-> enterprise session bean
VersionChecker Classloader Info

Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@541623cf
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestEJB.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Cla
ssloaderTest.ear\myjars\ClassloadersTest.jar;E:\WebSphereHandbooks\V51Handb
ookSource\Code\ClassloadersTestV2.jar;
Delegation Mode: PARENT_FIRST
Class version is: 1.0
```

Now, although we have put the V2.0 of the classloadersTest.jar on the shared library, we see V1.0 being loaded. This is because we are loading the ClassloadersTest.jar which is inside the EAR file (we have not removed it!). The list of JAR files you declare in a shared library is *appended* to the application classloader classpath. Keep in mind that *order is very important*.

Therefore, if you want the shared library code to be loaded, you will have to remove the `Class-path` entry from the EJB module MANIFEST.MF file, restart the application, and try again.

Using shared libraries at the application server level

A shared library can also be associated to an application server. All applications deployed in this server will "see" the code listed on that shared library. To associate a shared library to an application server, you must first create a classloader as follows:

1. Select an application server. In its Additional Properties list, select **Classloader**.
2. Choose **New**, and select a classloading policy for this classloader (PARENT_FIRST/PARENT_LAST). If you set the application policy to PARENT_LAST, you will be able to use your own classes in place of some WebSphere runtime classes for example (typically, XML parsers or transformers).
3. Select the **Libraries** entry.
4. Click **Add**, and select the library you want to associate to this application server. Repeat this operation if you want to associate multiple libraries to this classloader.
5. Click **OK**.
6. Save the configuration.

If we try to attach the VersionCheckerV2_SharedLib shared library to the application server, we obtain the following trace:

```
Direct call from Servlet to Version Checker Class

Class Version Checker was loaded by:
com.ibm.ws.classloader.CompoundClassLoader@272b6d98
Local ClassPath:
F:\WebSphere\AppServer51\installedApps\EAGLENetwork\ClassloaderTest.ear\Cla
ssloaderTestWeb.war\WEB-INF\classes;F:\WebSphere\AppServer51\installedApps\
EAGLENetwork\ClassloaderTest.ear\ClassloaderTestWeb.war\WEB-INF\lib\Classlo
adersTestV2.jar;F:\WebSphere\AppServer51\installedApps\EAGLENetwork\Classlo
aderTest.ear\ClassloaderTestWeb.war;
Delegation Mode: PARENT_LAST
Class version is: 2.0
```

```
Call the VersionChecker via Access bean-> enterprise session bean
VersionChecker Classloader Info

Class Version Checker was loaded by:
com.ibm.ws.classloader.ExtJarClassLoader@2053008795
Local ClassPath:
E:\WebSphereHandbooks\V51HandbookSource\Code\ClassloadersTestV2.jar;
Delegation Mode: PARENT_FIRST
Class version is: 2.0
```

Note that the WAR classloader continues to load its own version due to the delegation mode (PARENT_LAST) while the EJB module loads the code from the highest classloader in the hierarchy (the one we just defined at the application server level).

14.8 A special case: XML parsers and transformers

Those libraries are probably the number one source of issues with classloaders. WebSphere provides its own version of libraries, and it is very common for customers to use their own. The trick here is the delegation mode: as you saw in the previous example, it is not sufficient to place JAR files in the WEB-INF/lib folder of a WAR module or at the root of the EAR. Unless you use the `PARENT_LAST` delegation mode, the WAR classloader or application classloader will continue to delegate the loading of those classes to their parent classloader.

Wherever you place your own XML parsers and transformers, you will have to set the corresponding classloader delegation to `PARENT_LAST`. You can use the XalanVersionChecker.jsp provided with this sample to verify the version of the XML parsers and transformer your servlets code actually uses. We also highly recommend using the ClassloaderViewer tool to resolve such issues (see 14.11.1, “Using the Classloader Viewer Tool” on page 708).

Example 14-9 XalanVersionChecker Output

```
Xalan/Xerces Versions Checker
Xalan Version
XSLT4J Java 2.5.4
Xerces Version
XML4J 4.2.2
```

14.9 Handling JNI code

Due to a JVM limitation, code that needs to access native code via a Java Native Interface (JNI) must not be placed on a reloadable classpath, but on a static classpath. This includes shared libraries for which you can define a native classpath, or the application server classpath.

14.10 Migration from V4.0: considerations

Tip: Before migrating to Version 5.0, we recommend you run your application with the J2EE classloader policy introduced in V4.03. This policy can be set by using the `-Dcom.ibm.ws.classloader.J2EEApplicationMode= true` property on an application server.

The following table maps the 4.0x module visibility modes into their logical equivalents on WebSphere V5.0. WebSphere V5.0 provides additional flexibility, since some applications running in a server can be configured with a Web module classloader policy of Module, while others use an Application policy.

WebSphere V4.0 Mode	**Application Classloader Policy**	**WAR Classloader Policy**
Server	SINGLE	Application
Compatibility	SINGLE	Module
Application	MULTIPLE	Application
J2EE	MULTIPLE	Module

There is no exact equivalent to the Version 4.0 Module mode, since it provided isolation of EJB modules within an application, which is not possible at all in Version 5.0 (all EJB JAR files within an application are always loaded by the same classloader).

14.11 Troubleshooting

In this section, we discuss where to find information to help you resolve classloaders problems.

14.11.1 Using the Classloader Viewer Tool

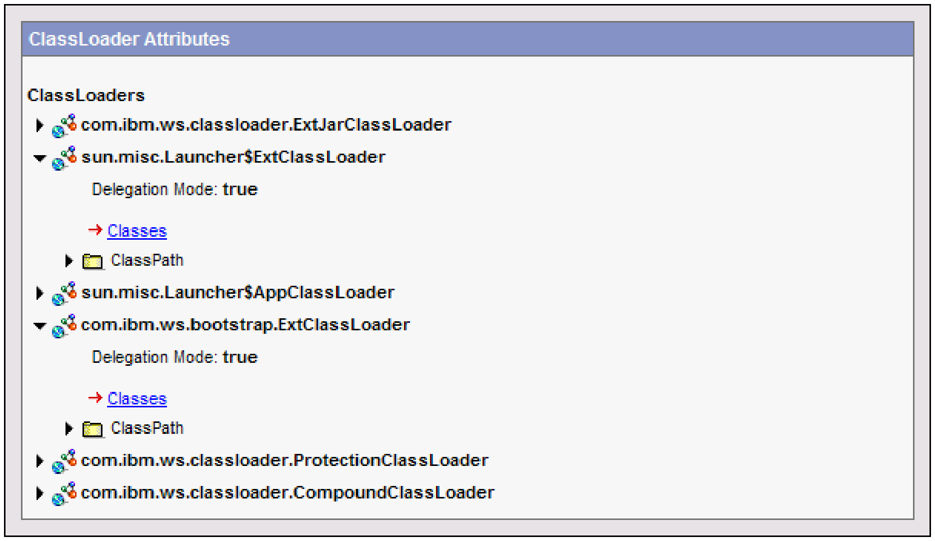
The Classloader Viewer Tool is an additional plug-in to the Web administrative console. It provides live information about classloaders, their hierarchy, and which classloader loaded which class. It can be very useful to debug classloader issues. This tool can be installed in a 5.0.x or a 5.1 console (base or ND). Details about the classloader viewer are available in the WebSphere Developer Domain, at:

`http://www.ibm.com/developerworks/websphere/library/techarticles/0312_cocasse/0312_cocasse.html`

Important: Please note that this tool is provided as is. It is a not part of the official WebSphere Application Server product.

You will be able to perform the following tasks using the classloader viewer:

- View the hierarchy of classloaders (as shown in Figure 14-22)
- List all classes loaded by a specific classloader
- Search for a JAR or class, and find on which classpath it is listed
- Save an XML file containing all pertinent classloaders information



ClassLoader Attributes

ClassLoaders

- com.ibm.ws.classloader.ExtJarClassLoader
- sun.misc.Launcher$ExtClassLoader
  - Delegation Mode: true
  - Classes
  - ClassPath
- sun.misc.Launcher$AppClassLoader
- com.ibm.ws.bootstrap.ExtClassLoader
  - Delegation Mode: true
  - Classes
  - ClassPath
- com.ibm.ws.classloader.ProtectionClassLoader
- com.ibm.ws.classloader.CompoundClassLoader

Figure 14-22 Classloaders hierarchy

14.11.2 Tracing classloaders

Alternatively, you can obtain a very detailed trace of what is happening in the system by setting the diagnosis trace to `Websphere ClassLoader=all=enabled`. This trace includes:

- Information on which classloaders are created, their classpaths, and their policy/delegation mode.
- Information on which modules have been detected.
- Information on shared libraries found.
- Information on dependency JARs found.
- Step-by-step information on the "search for classes" (where a class has been found and what was done to find it).

Example 14-10 WebSphere classloaders trace example

```
[11/14/02 2:00:41:248 CET] 124d22bf ClassGraph    d finished creating
classloaders:
this=com.ibm.ws.classloader.ClassGraph@33c6e2b7
singleServerClassLoader: false
singleWarClassLoader: false
modules          : 2
dependent paths  : 2
deppath:
F:\WebSphere\AppServer50\installedApps\EAGLENetwork\ClassloaderTest.ear\Classlo
aderTestWeb.war\JARS\ClassloadersTestV2.jar
deppath:
F:\WebSphere\AppServer50\installedApps\EAGLENetwork\ClassloaderTest.ear\Classlo
adersTest.jar
library paths  : 0
module: ClassloaderTestWeb.war
F:\WebSphere\AppServer50\installedApps\EAGLENetwork\ClassloaderTest.ear\Classlo
aderTestWeb.war\WEB-INF\classes
F:\WebSphere\AppServer50\installedApps\EAGLENetwork\ClassloaderTest.ear\Classlo
aderTestWeb.war\WEB-INF\lib\ClassloadersTestV2.jar
F:\WebSphere\AppServer50\installedApps\EAGLENetwork\ClassloaderTest.ear\Classlo
aderTestWeb.war
module: ClassloaderTestEJB.jar
F:\WebSphere\AppServer50\installedApps\EAGLENetwork\ClassloaderTest.ear\Classlo
aderTestEJB.jar
```

14.12 Dynamic and hot deployment

WebSphere Application Server V5 provides support for dynamic deployment, that is the ability to deploy new code without stopping the application. The “Hot deployment and dynamic reloading” article in the InfoCenter exhaustively details which dynamic changes can be made.

Note: The relationship between all the classloaders at runtime makes dynamic reloading sometimes difficult. Some classes will be reloaded but dependent classes might not be, which may lead to ClassCastExceptions. Static classes might be an issue as well. Your safer bet is to reload the complete application. You then have the guarantee that all related classes are reloaded in unison.

14.13 Separating static content from dynamic content

By default, WebSphere serves all the artifacts of an application, should they be static or dynamic. If your application uses a lot of static content, you may want to configure WebSphere so that the static content is served by the Web server instead. This comes at the expense of packaging the static contents of the application separately, and deploying them manually (there is no standard process to do this).

To do this, you must disable the file serving servlet. This can be done from the AAT, in the IBM Extensions tab of the Web Module Properties window, by un-checking the **File serving enabled** option.

You must then reinstall the application and regenerate the plug-in configuration. To check that this has worked, you can look at the plugin-cfg.xml file, where you should see something similar to this, as opposed to a single rule that redirects all `/webbank/*` requests to WebSphere:

```
<UriGroup Name="webbankApplication/webbankWeb_URIs">
   <Uri Name="/webbank/TransferServlet"/>
   <Uri Name="/webbank/*.jsp"/>
   <Uri Name="/webbank/*.jsv"/>
   <Uri Name="/webbank/*.jsw"/>
   <Uri Name="/webbank/j_security_check"/>
</UriGroup>
```

You should also configure the Web server to recognize the `/webbank` URI. This can easily be done in Apache/IHS using an Alias directive, like this:

```
Alias /webbank/* "D:/webbank/web/"
```

This assumes all the static content of the Webbank application has been deployed in the D:/webbank/web directory.

14.14 Maintenance best practices

The main piece of advice is to keep your EAR files in sync. It is likely that you will use a tool such as Rational® ClearCase® or Intersolv PVCS to manage your code. Basically, you should always be able to retrieve from your configuration management tool the exact configuration that runs in production or test.

You can get out of sync by updating code directly under the <WAS_HOME>/installedApps directory, mainly for hot deployment purposes, as opposed to redeploying an EAR file that contains the latest changes. If you do this, make sure you rebuild an EAR that contains the latest changes, and put this EAR file back in the configuration management system. However, the best way to update an application is definitely to use your configuration management system as the unique place you deploy from, and the EAR file as the deployment unit. This is especially true if you need to install an application on multiple systems, where maintenance can become a nightmare.

Part 5

Managing WebSphere

This part includes information on troubleshooting runtime problems and on using command line and scripting to manage WebSphere Application Server.

15

Troubleshooting

Problems within an e-business environment can take many forms, including poor performance, application unavailability, or unexpected results. The first step in problem resolution is to isolate the problem and understand what it is.

In this chapter, we introduce tools and techniques that can be used to analyze and correct problems. Included in this chapter is information on:

- Console messages, logs, and traces
- Log Analyzer
- Thread Analyzer technology preview and the Java stack trace
- Collector Tool
- First Failure Data Capture (FFDC) logs
- System core dump analysis
- Application Server Toolkit and JRas framework.

15.1 Resources for identifying problems

WebSphere provides the following sources of feedback to help with problem determination. Each are described separately in subsequent sections:

- **Administrative console messages**. These provide important information regarding runtime events and configuration problems. They are an important starting point to determine the cause of any configuration problem.
- **Log files**. Several general-purpose logs are provided, such as JVM standard logs, process (native) logs, and IBM service logs.
- **Traces**. Traces provide more detailed information about WebSphere components to determine what is wrong with your WebSphere environment.
- **Log Analyzer**.The Log Analyzer is a GUI tool that permits the user to view any logs generated with log analyzer trace format, such as the IBM service log file. This tool allows the user to get error message explanations and information such as why the error occurred and how to recover from it.
- **Thread Analyzer**. The Thread Analyzer technology preview assists analyzing Java stack trace files and viewing the threads, both for hung or deadlock conditions, or to simply evaluate application processing.
- **Collector tool**. The Collector tool gathers information about the application server installation and packages it in an output JAR file. The information in the file includes logs, property files, configuration files, operating system and Java data, and prerequisite software presence and levels. The file can be sent to IBM Customer Support to assist in problem determination and analysis.

  Beginning with V5.0.2, the Collector tool includes the -Summary option, which produces a lightweight text file and console version of some information from the JAR file produced without this option.
- **FFDC**. The First Failure Data Capture (FFDC) function preserves the information generated from a processing failure and returns control to the affected engines. The captured data is saved automatically for use in analyzing the problem, and could be collected by the Collector tool.

15.2 Administrative console messages

Runtime status messages are displayed at the WebSphere Status area at the bottom of the administrative console, providing information regarding runtime events and configuration problems.

The area consists of two frames: WebSphere Runtime Messages, seen in Figure 15-1 on page 717, and WebSphere Configuration Problems, seen in

Figure 15-2. You can toggle between the two by clicking the **Previous** and **Next** links.

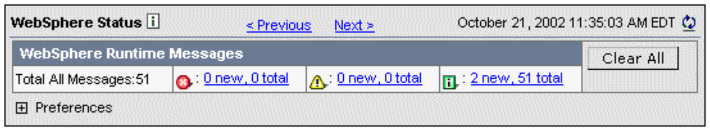


Figure 15-1 WebSphere status: runtime messages

Runtime messages can be cleared by clicking the **Clear All** button to the right of the message list.

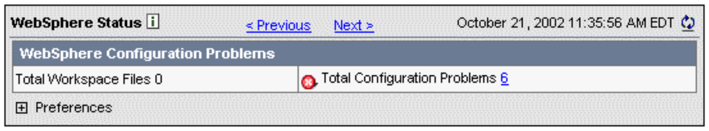
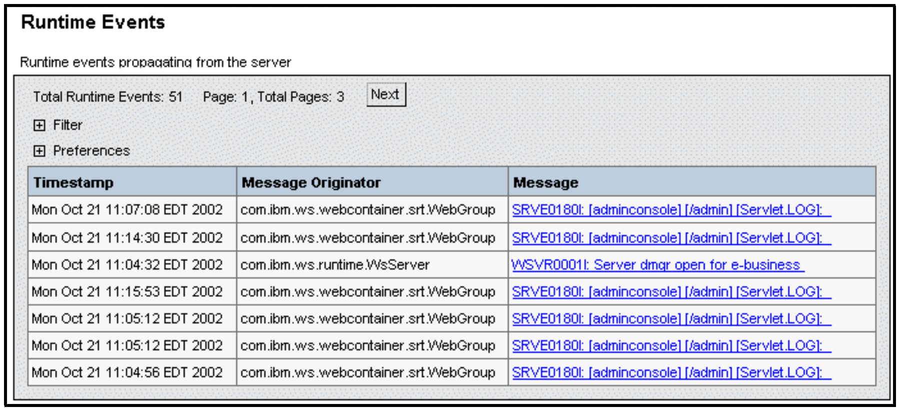

WebSphere Status
< Previous
Next >
October 21, 2002 11:35:56 AM EDT
WebSphere Configuration Problems
Total Workspace Files 0
Total Configuration Problems 6
Preferences

Figure 15-2 WebSphere status: configuration problems

Messages indicating configuration problems are cleared when the problem is corrected.

To view the messages, select the icon for the type of messages you want to view. Error messages are represented by the **x** icon, warning messages by the **!** icon, and informational messages by the **i** icon. Figure 15-3 shows the results of clicking the runtime information messages link in Figure 15-1.



Runtime Events

Runtime events propagating from the server

Total Runtime Events: 51 Page: 1, Total Pages: 3 Next

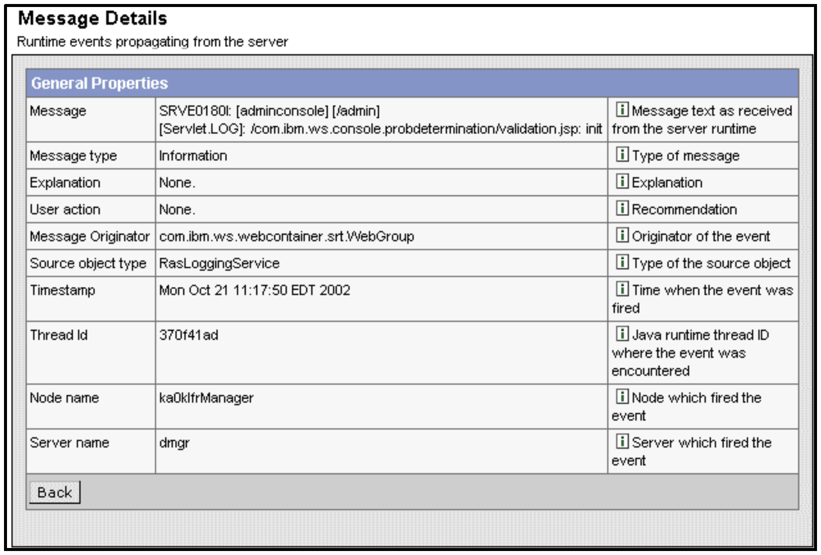
Filter

Preferences

Timestamp	Message Originator	Message
Mon Oct 21 11:07:08 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:14:30 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:04:32 EDT 2002	com.ibm.ws.runtime.WsServer	WSVR0001I: Server dmgr open for e-business
Mon Oct 21 11:15:53 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:05:12 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:05:12 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:
Mon Oct 21 11:04:56 EDT 2002	com.ibm.ws.webcontainer.srt.WebGroup	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]:

Figure 15-3 Runtime events

When the list of messages appears, you can view the details of the message by selecting the link in the Message column. Figure 15-4 shows an example of message details you might see.


Message Details

Runtime events propagating from the server

Message	SRVE0180I: [adminconsole] [/admin] [Servlet.LOG]: /com.ibm.ws.console.probdetermination/validation.jsp: init	Message text as received from the server runtime
Message type	Information	Type of message
Explanation	None.	Explanation
User action	None.	Recommendation
Message Originator	com.ibm.ws.webcontainer.srt.WebGroup	Originator of the event
Source object type	RasLoggingService	Type of the source object
Timestamp	Mon Oct 21 11:17:50 EDT 2002	Time when the event was fired
Thread Id	370f41ad	Java runtime thread ID where the event was encountered
Node name	ka0klfrManager	Node which fired the event
Server name	dmgr	Server which fired the event

Figure 15-4 Message details

15.3 Log files

WebSphere Application Server can write system messages to several general-purpose logs. These include:

- **JVM logs** are created by redirecting the System.out and System.err streams of the JVM. By default, these files are stored as <WAS_HOME>/logs/<server_name>/SystemOut.log and SystemErr.log.
- **Process (native) logs** are created by redirecting the stdout and stderr streams of the process's native module (.dlls, .so, UNIX libraries, and other JNI native modules), including the JVM native code itself. These logs can contain information relating to problems in native code or diagnostic information written by the JVM. By default, these files are stored as <WAS_HOME>/logs/<server_name>/native_stderr.log and native_stdout.log.
- **Service log** is a special log named, by default, activity.log, written in a binary format. You cannot view the log directly using a text editor. Log Analyzer or the Showlog tool is required to view the log.

15.3.1 JVM (standard) logs

The JVM (standard) logs are created by redirecting the System.out and System.err streams. WebSphere Application Server writes formatted messages to the System.out stream. In addition, applications and other code can write to these streams using the print() and println() methods defined by the streams. Some JDK built-ins such as the printStackTrace() method on the Throwable class can also write to these streams.

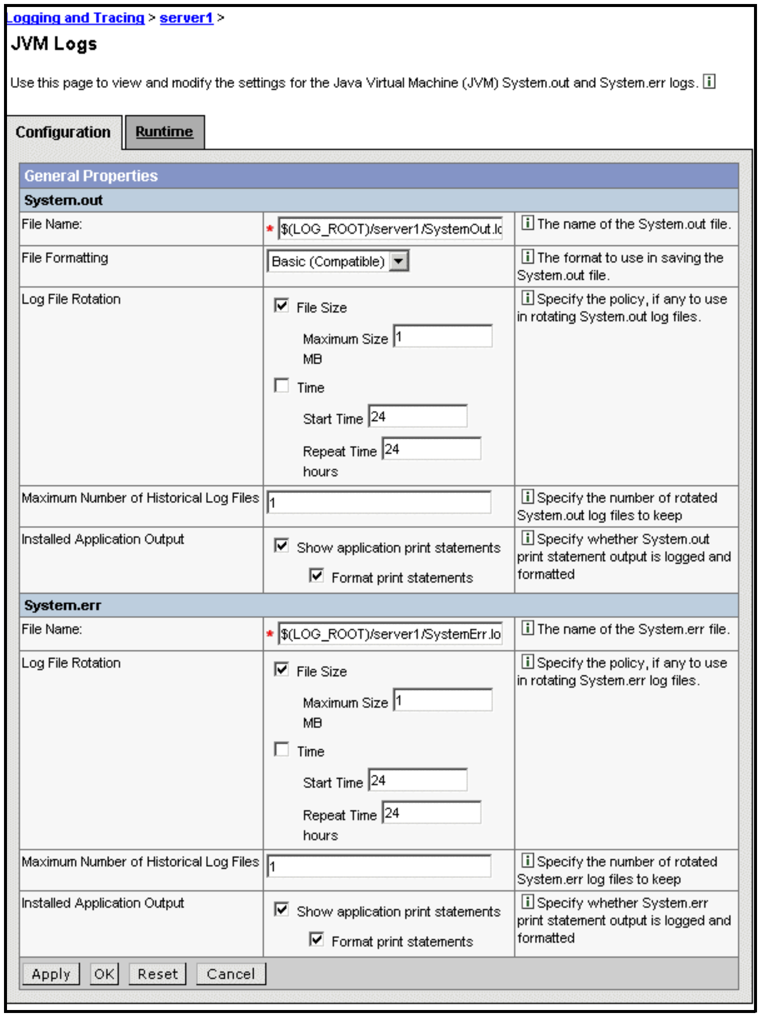
Typically, the System.out log is used to monitor the health of the running application server. The System.err log contains exception stack trace information that is useful when performing problem analysis.

Configuring the JVM logs

To view and modify the settings for the JVM System.out and System.err logs using the administrative consoles:

1. Click **Troubleshooting -> Logs and Trace** in the navigation tree.
2. Select a server by clicking the server name.
3. Click **JVM Logs.**
4. Select the **Configuration** tab.

5. Scroll through the window to display the attributes.


Logging and Tracing > server1 >

JVM Logs

Use this page to view and modify the settings for the Java Virtual Machine (JVM) System.out and System.err logs.

Configuration | Runtime

General Properties

System.out

File Name:	* $(LOG_ROOT)/server1/SystemOut.lc	The name of the System.out file.
File Formatting	Basic (Compatible)	The format to use in saving the System.out file.
Log File Rotation	☑ File Size Maximum Size 1 MB ☐ Time Start Time 24 Repeat Time 24 hours	Specify the policy, if any to use in rotating System.out log files.
Maximum Number of Historical Log Files	1	Specify the number of rotated System.out log files to keep
Installed Application Output	☑ Show application print statements ☑ Format print statements	Specify whether System.out print statement output is logged and formatted

System.err

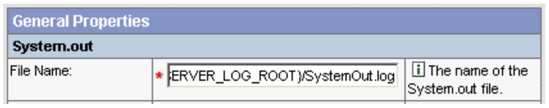
File Name:	* $(LOG_ROOT)/server1/SystemErr.lo	The name of the System.err file.
Log File Rotation	☑ File Size Maximum Size 1 MB ☐ Time Start Time 24 Repeat Time 24 hours	Specify the policy, if any to use in rotating System.err log files.
Maximum Number of Historical Log Files	1	Specify the number of rotated System.err log files to keep
Installed Application Output	☑ Show application print statements ☑ Format print statements	Specify whether System.err print statement output is logged and formatted

Figure 15-5 Configuring the JVM logs setting

- **File Name:**

  Specifies the name of the System.out or System.err file. The file name specified on the Configuration tab must have one of the following values:

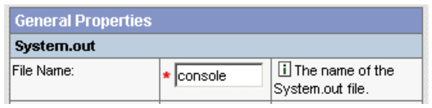
  – **filename**: The name of a file in the file system.You can use a fully qualified file name. If the file name is not fully qualified, it is considered to be relative to the current working directory (<WAS_HOME>) for the server. The file will be created if it does not exist.


General Properties

System.out

File Name: * ERVER_LOG_ROOT}/SystemOut.log — The name of the System.out file.

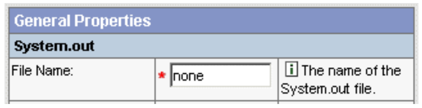
– **console**: This is a special file name used to redirect the stream to the corresponding process stream. If this value is specified for System.out, the file is redirected to stdout. If this value is specified for System.err, the file is redirected to stderr.


General Properties

System.out

File Name: * console — The name of the System.out file.

– **none**: Discards all data written to the stream. Specifying none is equivalent to redirecting the stream to dev/null on a UNIX system.


General Properties

System.out

File Name: * none — The name of the System.out file.

- **File formatting:**

  Specifies the format to use in saving the System.out file:

  – **Basic:** format used in earlier versions of WebSphere Application Server. Basic is the default.

  <timestamp><threadID><shortName><eventType>[class][method]<mess age>

  – **Advanced:** format extending the basic format by adding information about an event, when possible.

  <timestamp><threadID><eventType><UOW><source=longName>[class][method]<Organization><Product><Component><message>

  The addition of the unit of work information is particularly valuable when debugging in a distributed environment.

- **Log file rotation:**

  A self-managing log file will write messages to a file until some criteria, either size or time, is reached. At the specified time or when the file reaches the specified size, the current file is closed and renamed to a name consisting of the current name plus a timestamp. The stream then reopens a new file reusing the original name and continues writing.

  – File size

    If this option is selected, the file automatically performs self-maintenance by rolling over the file when it reaches the specified maximum size.

  – Maximum size

    This attribute specifies the maximum size in megabytes to which the file is allowed to grow.

  – Time

    Selecting this attribute allows the log file to manage itself based on the age of the file. If this option is selected, the file will roll itself over after the specified time period.

  – Start time

    Specifies the hour of the day, from 1 to 24, from which the periodic rollover algorithm commences. The periodic rollover algorithm uses this hour to load the algorithm at application server startup. Once started, the rollover algorithm runs without adjustment until the application server is stopped.

  – Rollover period

    Specifies the number of hours after which the log file will be rolled over to a new file name. Valid values are from 1 to 24.

    Note that if both file size and time are selected, the file is rolled over based on the criteria met first.

- **Maximum Number of Historical Log Files:**

  Specifies the number of rolled-over files to keep.

- **Installed Application Output:**

  Specifies whether the System.out or System.err print statements issued from the application code are logged and formatted.

  – Show application print statements

    Causes application messages written to this stream using the print and println stream methods to be shown. This will have no effect on system messages written to the stream by the WebSphere Application Server.

  – Format print statements

    Causes application messages written to this stream using the print and println stream methods to be formatted like WebSphere system messages.

6. Change the appropriate configuration attributes and click **Apply**.

7. Save your configuration changes.

The JVM logs are written as plain text files, so you can open and view the files directly using your own editor. You can also view the JVM logs using the Runtime tab as shown in Figure 15-5 on page 720, enabling viewing the JVM logs from a remote machine.

JVM log message formats

Analyzing and understanding logs is a significant step in the problem determination process. Messages logged by application server components and associated IBM products start with a unique message identifier that indicates the component or application that issued the message.

Example 15-1 illustrates the basic format of a log or trace entry.

Example 15-1 JVM logs (basic format)

```
[10/16/02 13:37:35:516 EDT] 50e907e8 WebContainer  E SRVE0146E: Failed to Start
Transport on host , port 9090. The most likely cause is that the port is
already in use. Please ensure that no other applications are using this port
and restart the server. com.ibm.ws.webcontainer.exception.TransportException:
Failed to start transport http: java.net.BindException: Address in use: bind
```

A description of each field is shown in Table 15-1.

Table 15-1 Log entry format description

Field	Example	Description
Time stamp	`[10/16/02 13:37:35:516 EDT]`	The time stamp in fully qualified date, time and time zone format
Thread ID	`50e907e8`	The thread ID or the hash code of the thread issuing this message
Component	`WebContainer`	The short name of component issuing this message

Field	Example	Description
Event Type	`E`	The type of the message or trace event. Possible values include: **A** Audit **I** Informational **W** Warning **E** Error **F** Fatal **O** System.out by the user application or internal components **R** System.err by the user application or internal components **u** A special type used by the message logging component of the WebSphere Application Server runtime **Z** A placeholder to indicate the type was not recognized
Message ID	`SRVE0146E`	The identifier of the message.
Message	`Failed to Start Transport on host , port 9090. The most likely cause is that the port is already in use. Please ensure that no other applications are using this port and restart the server.`	The text of the message and message arguments

The message identifier (message ID) can be either 8 or 9 characters in length and has the form:

```
CCCC1234X
```

Where

- `CCCC` is a four-character alphabetic component or application identifier.
- `1234` is a four-character numeric identifier used to identify the specific message for that component.
- `X` is an optional alphabetic severity indicator (I = Informational, W = Warning, E = Error)

To view the message IDs or the meaning of the messages generated by WebSphere Application Server components, select the **Quick Reference** view on the InfoCenter and expand the **Messages** topic.

15.3.2 Process (native) logs

The stdout and stderr streams written by native modules (.dlls, .so, UNIX libraries, and other JNI modules) are redirected to the native log files at application server startup. By default, these files are stored as <WAS_HOME>/logs/<server_name>/native_stderr.log and native_stdout.log.

To view or change the log settings or to view the log:

1. Click **Servers -> Application Servers**.
2. Select the server by clicking the name.
3. Select **Process Definition** in the Additional Properties table.
4. Select **Process Logs** in the Additional Properties table.
5. To view the settings, select the **Configuration** tab. To view the logs select the **Runtime** tab.

Note: This is a change from previous versions of WebSphere Application Server, which by default had one log file for both JVM standard output and native standard output, and one log file for both JVM standard error and native error output.

15.3.3 IBM service (activity) log

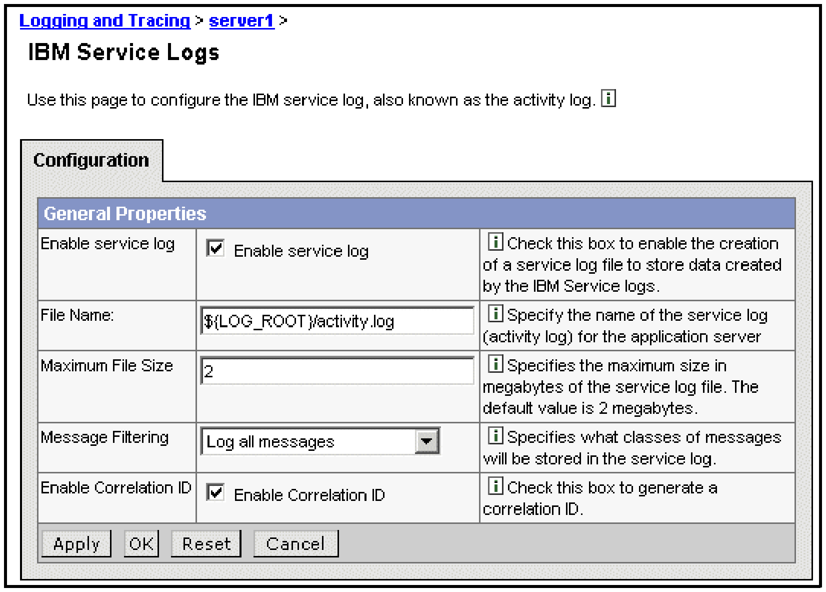
The IBM service log is a special log written in a binary format that captures events that show a history of WebSphere Application Server activities, also known as the *activity log*.

By default, the IBM service log is shared among all server processes for a node. The configuration values for the IBM service log are inherited by each server process from the node configuration.

Note: You can configure a separate IBM service log for each server process by overriding the configuration values at the server level.

Follow these steps to view or change the IBM service log settings using the administrative console:

1. Select **Troubleshooting -> Logs and Trace**.
2. Select the server by clicking the name.
3. Select **IBM Service Logs**.


Logging and Tracing > server1 >

IBM Service Logs

Use this page to configure the IBM service log, also known as the activity log.

Configuration

General Properties

Enable service log	✓ Enable service log	Check this box to enable the creation of a service log file to store data created by the IBM Service logs.
File Name:	${LOG_ROOT}/activity.log	Specify the name of the service log (activity log) for the application server
Maximum File Size	2	Specifies the maximum size in megabytes of the service log file. The default value is 2 megabytes.
Message Filtering	Log all messages	Specifies what classes of messages will be stored in the service log.
Enable Correlation ID	✓ Enable Correlation ID	Check this box to generate a correlation ID.

Figure 15-6 IBM Service log

- Check the **Enable** box to enable the service log or clear the check box to disable the log.
- **File Name:** Set the name for the service log. The default name is activity.log. If the name is changed, the runtime requires write access to the new file, and the file must use the .log extension.
- **Maximum File Size:** Specifies the number of megabytes to which the file can grow. When the file reaches this size, it wraps, replacing the oldest data with the newest data.
- **Message Filtering:** Set the message filter level to the desired state. You can select to store all messages or select a combination of service, warning, and error message.
- **Enable Correlation ID:** This option allows you to specify whether a correlation ID should be generated and included in message events and diagnostic trace entries. If set to `true`, each application client request is assigned a unique identifier that is propagated to all servers touched as part of servicing that request. This allows correlation of events across multiple server processes.

4. Save the configuration.
5. Restart the server to apply the configuration changes.

To view the service log, use the **showlog** command in the <WAS_HOME>/bin directory. This command dumps the binary log file to standard out or a file. The format is:

```
showlog binaryFilename [outputFilename]
```

`outputFilename` is optional. If no file name is given, **showlog** dumps the service log file to standard out. For example:

```
showlog ../logs/activity.log
```

Another powerful way to view the service log file is to use Log Analyzer (described later in 15.5, “Log Analyzer” on page 736).

15.4 Traces

Tracing can be useful if you have problems with particular components of WebSphere Application Server, clients and other processes, and the log files don’t provide you with enough information to determine the problem.

Note: Traces need manual activation, and you need to remember to turn off tracing when you are done collecting the information. Traces may impact performance rather severely.

Tracing is most likely to be used by IBM Service for diagnosing WebSphere problems and not by the typical user diagnosing application problems.

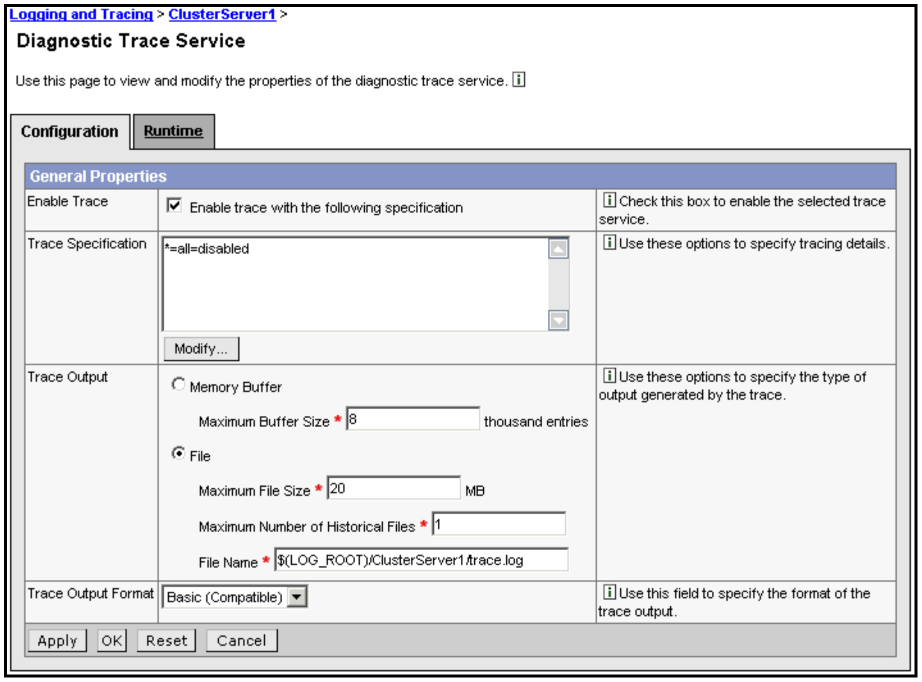
15.4.1 Diagnostic trace service

By default, the trace for all WebSphere Application Server components is disabled.

Enabling trace at server startup

The trace configuration settings are read at server startup time and used to configure the trace service. To configure or change the diagnostic trace settings, using the administrative console:

1. Select **Troubleshooting -> Logs and Trace**.
2. Select the server by clicking the name. You can select an application server, node agent, JMS server, or the Deployment Manager.
3. Select **Diagnostic Trace**.
4. Select the **Configuration** tab.


Logging and Tracing > ClusterServer1 >

Diagnostic Trace Service

Use this page to view and modify the properties of the diagnostic trace service.

Configuration | Runtime

General Properties

Enable Trace	☑ Enable trace with the following specification	Check this box to enable the selected trace service.
Trace Specification	*=all=disabled Modify...	Use these options to specify tracing details.
Trace Output	○ Memory Buffer Maximum Buffer Size * 8 thousand entries ● File Maximum File Size * 20 MB Maximum Number of Historical Files * 1 File Name * $(LOG_ROOT)/ClusterServer1/trace.log	Use these options to specify the type of output generated by the trace.
Trace Output Format	Basic (Compatible)	Use this field to specify the format of the trace output.

Figure 15-7 Trace service configuration

- Check the **Enable Trace** box to enable tracing.
- **Trace Specification:** Set the proper trace strings. The options for this are described in “Trace string specification” on page 729.
- **Trace Output:** Select whether to direct trace output to either a file or an in-memory circular buffer.
  - If the in-memory buffer is selected, set the size of the buffer, specified in thousands of lines. When using the in-memory circular buffer, the buffer must be dumped to a file before it can be viewed. This can be done using the **Dump** button on the Runtime tab.
  - If a file is selected, set the maximum size in megabytes to which the file should be allowed to grow. When the file reaches the size, the existing file will be closed, renamed and a new file with the original name reopened.
- **Trace output format:** Select the desired format for the generated trace. The options are Basic (Compatible) for a minimum trace, Advanced for detailed traces, and Log Analyzer.

5. Save the changed configuration.

6. Start (or restart) the server.

Note: You can also enable a trace for a running server. See "Enabling trace on a running server" on page 730.

Trace string specification

Trace strings must conform to a specific grammar for processing by the trace service:

```
COMPONENT_TRACE_STRING[:COMPONENT_TRACE_STRING]*
```

For example:

```
COMPONENT_TRACE_STRING = COMPONENT_NAME=LEVEL=STATE[,LEVEL=STATE]*
```

Where:

- `COMPONENT_NAME` is the name of a component or group registered with the trace service. Typically, WebSphere Application Server components register using a fully qualified Java class name, for example `com.ibm.servlet.engine.ServletEngine`. In addition, you can use a wildcard character of asterisk (*) to terminate a component name and indicate multiple classes or packages. For example, use a component name of `com.ibm.servlet.*` to specify all components whose names begin with `com.ibm.servlet`.
- `LEVEL = all | entryExit | debug | event`

  Level is the type of tracing to perform. You can specify more than one type by separating them with commas.
- `STATE = enabled | disabled`

Examples of legal trace strings include:

```
com.ibm.ejs.ras.ManagerAdmin=debug=enabled
com.ibm.ejs.ras.ManagerAdmin=all=enabled,event=disabled
com.ibm.ejs.ras.*=all=enabled
com.ibm.ejs.ras.*=all=enabled:com.ibm.ws.ras=debug=enabled,entryexit=enabled
```

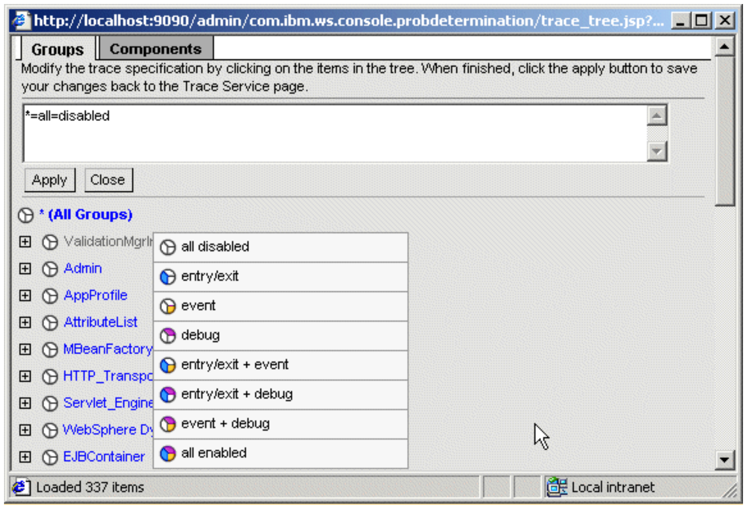
Trace strings cannot contain blanks and are processed from left to right. Specifying a trace string like:

```
abc.*=all=enabled,event=disabled
```

Enables the trace for all components whose names start with abc, then disables event tracing for those same components. This means that the trace string `abc.*=all=enabled,event=disabled` is equivalent to `abc.*=debug=enabled,entryexit=enabled`.

Graphical trace interface

An alternative to entering the trace string directly is to use the graphical trace interface to generate the trace strings that specifies the components, packages, or groups to trace. Click the **Modify** button (see Figure 15-7 on page 728) to start the graphical trace interface.



http://localhost:9090/admin/com.ibm.ws.console.probdetermination/trace_tree.jsp?...
Groups
Components

Modify the trace specification by clicking on the items in the tree. When finished, click the apply button to save your changes back to the Trace Service page.

```
*=all=disabled
```

Figure 15-8 Graphical trace interface

Enabling trace on a running server

You can also trace a server that is already active:

1. Select **Troubleshooting -> Logs and Trace**.
2. Select the server by clicking the name. You can select an application server, node agent, JMS server, or the Deployment Manager.
3. Select **Diagnostic Trace.**
4. Select the **Runtime** tab. The options are similar to those in the Configuration tab, described in “Enabling trace at server startup” on page 727.
   – Select the **Save Trace** check box if you want to write your changes back to the server configuration. If this option is not selected, the changes you make will apply only for the life of the server process that is currently running.
   – Specify the trace string as described in “Trace string specification” on page 729.
   – Configure the trace output. If using the in-memory buffer, the Dump button is used to dump the buffer to an existing file. This is necessary before viewing the trace output.

5. Click **Apply**.

Looking at trace output

Traces in basic format have the following format:

Example 15-2 Basic format

```
<timestamp><threadId><shortName><eventType>[className][methodName]<textmessage>
               [parameter 1]
               [parameter 2]
```

Traces in advanced format will have the following format:

Example 15-3 Advanced format

```
<timestamp><threadId><eventType><UOW><source=longName>[className][methodName]<O
rganization><Product><Component><textMessage>[parameter
1=parameterValue][parameter 2=parameterValue]
```

One of the most interesting fields is the EventType field. This is a one-character field in lowercase that indicates the type of the trace event. Possible values are:

```
> method entry
< method exit
e event
d debug
m dump
u unconditional
Z type was not recognized
```

Example 15-4 on page 732 shows the trace output in basic format.

Example 15-4 Trace output (basic format)

```
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d jspUri: /jvmlogs.jsp,
lastCheck (last time checked): 0, cachedJSPLastModTimestamp: 0, firstTime:
true, reloadEnabled: true, reloadInterval: 3000, debugEnabled: false
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d Inside main compile/reload
block.  firstTime: true, reload interval exceeded: true
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d  creating jsw.outDir
[11/11/02 16:40:27:625 EST] 1c5907d1 JspServlet    d  jsw.outDir:
C:\WebSphere\AppServer\temp\ka0klfr1\server1\DefaultApplication\DefaultWebAppli
cation.war\
...
[11/11/02 16:40:27:656 EST] 1c5907d1 JspServlet    d  loaded class:
org.apache.jsp._jvmlogs
[11/11/02 16:40:27:672 EST] 1c5907d1 HttpJspBase   > init
[11/11/02 16:40:27:672 EST] 1c5907d1 WebGroup      I SRVE0180I: [Default Web
Application] [/] [Servlet.LOG]: /jvmlogs.jsp: init
[11/11/02 16:40:27:672 EST] 1c5907d1 HttpJspBase   < init
```

15.4.2 Web server logs and traces

If a problem is suspected with the Web server or between the Web server and the Web container, there are several tools you can use.

Web server plug-in generation

The Web server plug-in configuration file controls what content is transferred from the Web server to an application server. This file must be regenerated after certain changes to the WebSphere configuration server and then moved to the proper location on the Web server.

If there is a problem with requests being routed to WebSphere Application Server from the Web server, make sure the Web server plug-in has been properly generated and moved to the Web server. For information on how to do this, see 8.6.8, “Generating the Web server plug-in” on page 330.

Web server plug-in log

The Web server plug-in creates a log file containing error and informational messages. The level of information placed in this log is determined by a setting in the Web server plug-in configuration file. Possible values in order of significance are Trace, Warn and Error (default). You can specify one value for the level. With Trace, you also get Warn and Error. Setting the level to Warn gives you Warn and Error.

Example 15-5 Web server plug-in configuration file log setting

```
<?xml version="1.0"?>
<Config>
    <Log LogLevel="Trace" Name="...\AppServer\logs\http_plugin.log"/>
    ...
</Config>
```

This setting has to be changed manually with a text editor. However, note that when you generate the Web server plug-in configuration using the administrative console, the log level will be replaced with the default value of Error.

> **Note:** Be careful when setting the level to Trace. A lot of error messages are logged at this level that can cause the disk space to fill up very quickly. A Trace setting should never be used in a normally functioning environment, because it affects performance.

IBM HTTP Server logs

The IBM HTTP Server has the following log files that aid in problem diagnosis:

- Error log
- Access logs

Error log

The error log records IBM HTTP Server errors. The location for the error log is specified with the ErrorLog directive. The LogLevel directive determines the level of logging. You can specify one of the following values for LogLevel:

emerg:	Emergencies - system is unusable.
alert:	Action must be taken immediately.
crit:	Critical conditions.
error:	Error conditions.
warn:	Warning conditions.
notice:	Normal but significant condition.
info:	Informational.
debug:	Debug level messages

When a particular level is specified, messages from all other levels of higher significance will be reported as well. A level of at least “crit” is recommended.

Access log

The access log records all Web server activity, including the following information for each request:

- What was requested
- Who requested it

- When it was requested
- The method used
- The type of file sent in response
- The return code

Often, the access log is combined with two other logs called the referrer and agent logs by specifying a log format that includes all the information normally found in each log. The information in these last two logs is of more interest to a webmaster who is gathering statistical information.

The format and location of the logs is determined by the log settings in the <IHS_HOME>/conf/httpd.conf configuration file. Example 15-6 shows the settings for the log files.

The LogFormat directives define the content of the log records. At the end of each LogFormat directive is a name identifying the format. Looking at the LogFormat directives in Example 15-6, the names are “combined”, “common”, “referer”, and “agent”. If you would like to customize the log formats, refer to the Apache Directives section of the IBM HTTP Server online documentation (under “Tell me about” / “Directives”).

The CustomLog directive indicates that the “custom” log format is to be used to store the log records. These records will be stored in /usr/IBMHttpServer/logs/access_log.

Example 15-6 IHS configuration file, httpd.conf

```
...
# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog /usr/IBMHttpServer/logs/error_log

# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.

LogLevel warn

# The following directives define some format nicknames for use with
# a CustomLog directive (see below).

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

```
# The location of the access logfile (Common Logfile Format).
# If this does not start with /, ServerRoot is prepended to it.

CustomLog /usr/IBMHttpServer/logs/access_log common
...
```

Including elapsed times in the log records

The default LogFormat record used ("common") doesn't include the setting for service elapsed time. The elapsed time is often useful in understanding performance problems. To add the elapsed time option, add %T at the LogFormat entry. This will report the time taken to serve the request, in seconds.

Example 15-7 Elapsed time option %T

```
...
#LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%h %l %u %t \"%r\" %>s %b %T" common
...
```

The elapsed time is reported in seconds.

IBM HTTP Server log rotation

On even a moderately busy server, the quantity of information stored in the log files is very large. It will consequently be necessary to periodically rotate the log files by moving or deleting the existing logs. This cannot be done while the server is running, because the server will continue writing to the old log file as long as it holds the file open. Instead, the server must be restarted after the log files are moved or deleted so that it will open new log files.

IBM HTTP Server is capable of writing error and access log files through a pipe to another process, rather than directly to a file. A simple example using piped logs:

```
# compressed logs
CustomLog "|/usr/bin/gzip -c >> /var/log/access_log.gz" common
# almost-real-time name resolution
CustomLog "|/usr/local/apache/bin/logresolve >> /var/log/access_log" common
```

One important use of piped logs is to allow log rotation without having to restart the server. The IBM HTTP Server includes a simple program called rotatelogs for this purpose. For example, to rotate the logs every 24 hours, you can use:

```
CustomLog "|/usr/local/apache/bin/rotatelogs /var/log/access_log 86400" common
```

ignore unnecessary log records in the access log file

Like all Web servers, the IBM HTTP Server records all HTTP access traffic in a log file. To manage the amount of information recorded, you can configure the log to ignore certain entries. Example 15-8 shows how to code the directives so that log entries for .gif and .jpg files are not recorded.

Example 15-8 Ignoring image entries in the access_log

```
...
SetEnvIf Request_URI \.gif$ ignore=gif
SetEnvIf Request_URI \.jpg$ ignore=jpg

#CustomLog /usr/IBMHttpServer/logs/access_log common
CustomLog /usr/IBMHttpServer/logs/access_log common env=!ignore
...
```

15.5 Log Analyzer

The Log Analyzer is a GUI tool that permits the user to view service and activity logs. It can take one or more logs, merge all the data, and display the entries in sequence.

More importantly, this tool is shipped with an XML database, the *symptom database*, which contains strings for some common problems, reasons for the errors, and recovery steps. The Log Analyzer compares every error record in the log file to the internal set of known problems in the symptom database and displays all the matches. This allows the user to get error message explanations and information such as why the error occurred and how to recover from it, as shown in Figure 15-9 on page 737.

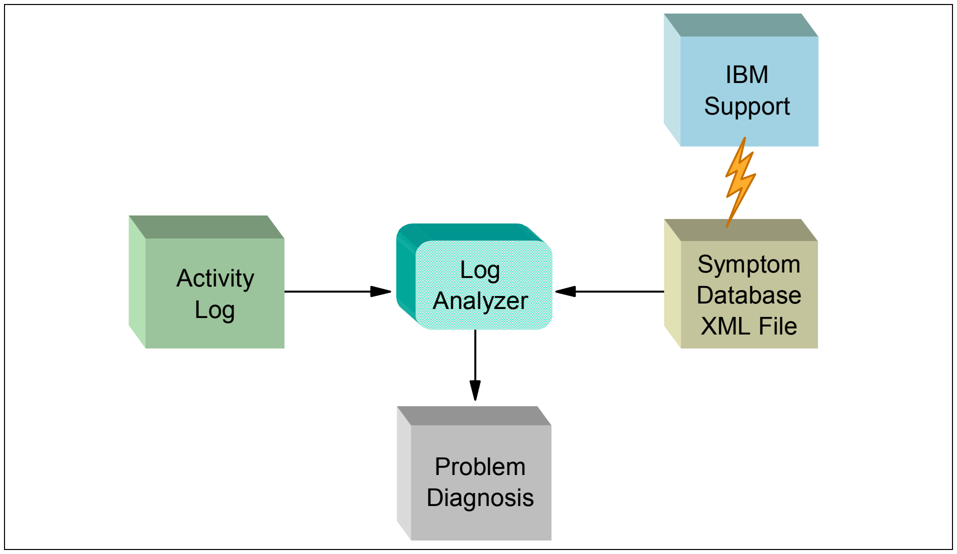


Figure 15-9 Log Analyzer concept

Note: The symptom database is maintained by IBM. We recommend that you refresh your copy often. To do this, see "Updating the symptom database" on page 743.

15.5.1 Using Log Analyzer

To start using the Log Analyzer:

1. Select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Log Analyzer** from the Windows start menu. If you are running from a Network Deployment system, select **Start -> Programs -> IBM WebSphere -> Application Server V5.1 -> Network Deployment -> Log Analyzer**.

   The Log Analyzer can also be started from the <WAS_HOME>/bin directory using the **`waslogbr.bat (.sh)`** command.

2. When the Log Analyzer GUI starts, select **File -> Open** from the main menu. Navigate to the <WAS_HOME>/logs directory, select **activity.log** and click **Open**.

You should now see the open activity log, similar to Figure 15-10 on page 738. All servers write log records to this file.

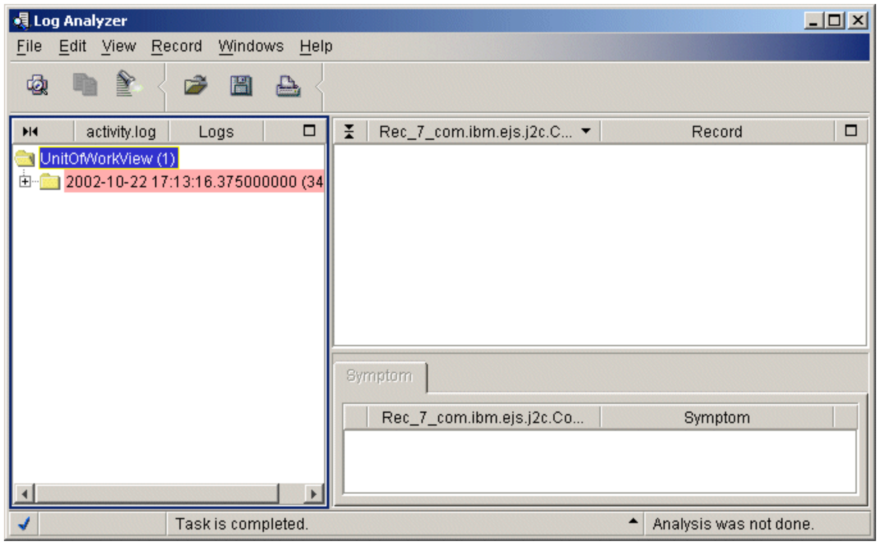
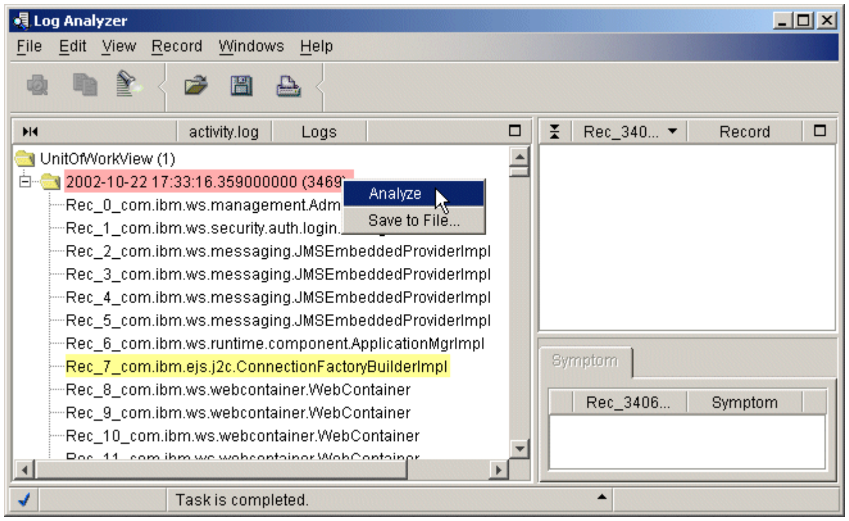


Figure 15-10 Log Analyzer

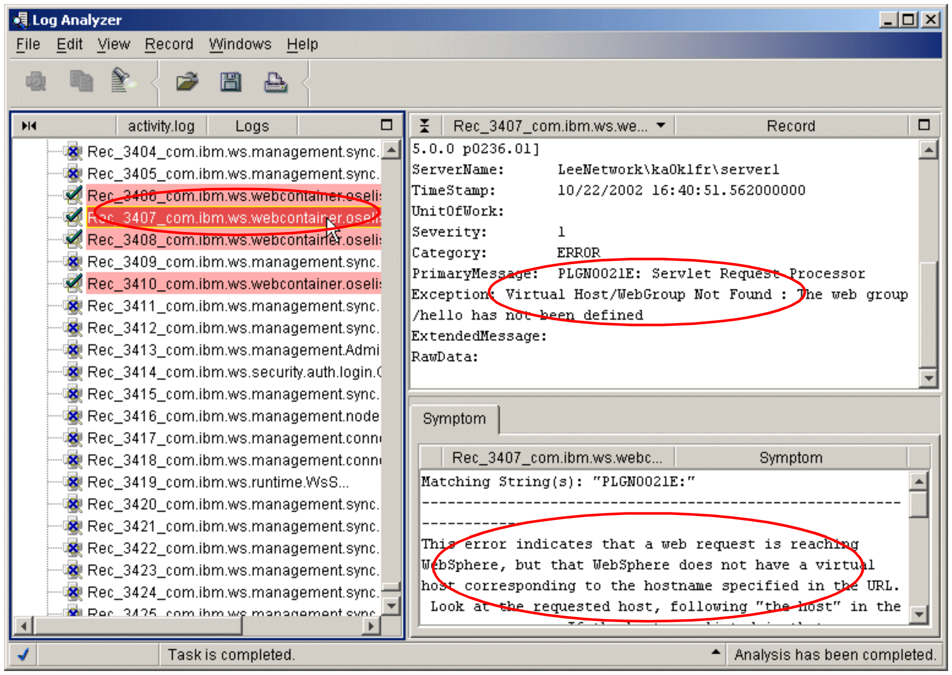
3. Select an entry in the UnitOfWorkView folder and the details will appear in the upper-right pane.
4. To analyze a log entry, right-click the entry and select **Analyze** from the pop-up menu, as shown in Figure 15-11 on page 739. The entry will be compared to the symptom database. If there is a match, the information will appear in the lower-right pane.



```
UnitOfWorkView (1)
2002-10-22 17:33:16.359000000 (3469
Rec_0_com.ibm.ws.management.Adm
Rec_1_com.ibm.ws.security.auth.login.
Rec_2_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
Rec_3_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
Rec_4_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
Rec_5_com.ibm.ws.messaging.JMSEmbeddedProviderImpl
Rec_6_com.ibm.ws.runtime.component.ApplicationMgrImpl
Rec_7_com.ibm.ejs.j2c.ConnectionFactoryBuilderImpl
Rec_8_com.ibm.ws.webcontainer.WebContainer
Rec_9_com.ibm.ws.webcontainer.WebContainer
Rec_10_com.ibm.ws.webcontainer.WebContainer
```

Figure 15-11 Selecting an entry to analyze

After the analyze action has been invoked, each analyzed log entry has an icon indicating whether analysis information is available. The check icon to the left of the entry in Figure 15-12 on page 740 indicates that analysis information is available.



```
Rec_3404_com.ibm.ws.management.sync.
Rec_3405_com.ibm.ws.management.sync.
Rec_3406_com.ibm.ws.webcontainer.oseli
Rec_3407_com.ibm.ws.webcontainer.oseli
Rec_3408_com.ibm.ws.webcontainer.oseli
Rec_3409_com.ibm.ws.management.sync.
Rec_3410_com.ibm.ws.webcontainer.oseli
Rec_3411_com.ibm.ws.management.sync.
Rec_3412_com.ibm.ws.management.sync.
Rec_3413_com.ibm.ws.management.Admi
Rec_3414_com.ibm.ws.security.auth.login.(
Rec_3415_com.ibm.ws.management.sync.
Rec_3416_com.ibm.ws.management.node
Rec_3417_com.ibm.ws.management.conn
Rec_3418_com.ibm.ws.management.conn
Rec_3419_com.ibm.ws.runtime.WsS...
Rec_3420_com.ibm.ws.management.sync.
Rec_3421_com.ibm.ws.management.sync.
Rec_3422_com.ibm.ws.management.sync.
Rec_3423_com.ibm.ws.management.sync.
Rec_3424_com.ibm.ws.management.sync.
Rec_3425_com.ibm.ws.management.sync.
```

```
5.0.0 p0236.01]
ServerName:       LeeNetwork\ka0klfr\server1
TimeStamp:        10/22/2002 16:40:51.562000000
UnitOfWork:
Severity:         1
Category:         ERROR
PrimaryMessage:   PLGN0021E: Servlet Request Processor
Exception: Virtual Host/WebGroup Not Found : The web group
/hello has not been defined
ExtendedMessage:
RawData:
```

```
Matching String(s): "PLGN0021E:"
------------------------------------------------------------
----------
This error indicates that a web request is reaching
WebSphere, but that WebSphere does not have a virtual
host corresponding to the hostname specified in the URL.
 Look at the requested host, following "the host" in the
```

Figure 15-12 Log Analyzer information

Log entries (left pane)

By default, the pane on the left displays log entries by unit of work (UOW). It lists all the UOW instances and associated entries from the logs that you have opened. You may find the UOW grouping useful when you are trying to find related entries in the service or activity log or when you are diagnosing problems across multiple machines.

The file name of the first log that you opened is shown in the pane's title bar. There is a root folder and under it, each UOW has a folder icon that you can expand to show all the entries for that UOW. All log entries without any UOW identification are grouped into a single folder in this tree view.

The UOW folders are sorted to show the UOW with the latest timestamp at the top of the list. The entries within each UOW are listed in the reverse sequence, that is the first (earliest) entry for that UOW is displayed at the top of the list. If you have merged several logs, all the log entries are merged in timestamp sequence within each UOW folder, as if they all came from the same log.

Each UOW line has the following format (see Figure 15-10 on page 738):

```
2002-10-22 17:33:16.359000000 (3469) 1-1880:itsohost/WebbankServer01
```

Where:

- `2002-10-22 17:33:16.359000000` is the timestamp.
- `(3469)` is the number of entries in the UOW.
- `1-1880:itsohost/WebbankServer01` is the unit of work.

Click the + icon next to the UOW folder to expand the folder (see Figure 15-12 on page 740). Each log entry's identification has the following format:

```
Rec_3407_com.ibm.ws.webcontainer.oselistener.OSEListenerDispatcher
```

Where:

- `Rec_3407` is the entry number.
- `com.ibm.ws.webcontainer.oselistener.OSEListenerDispatcher` is the class name.

Every log entry is assigned an entry number, `Rec_`*nnnn*, when a log is opened in the Log Analyzer. If more than one file is opened (merged files), the `Rec_`*nnnn* identification will not be unique because the number is relative to the entry sequence in the original log file and not to the merged data that is displayed. This `Rec_`*nnnn* also appears in the first line in the Records pane.

By default, each entry in this pane is color-coded to help you quickly identify the ones that have high severity errors.

Non-selected log entries have a background color of:

- Pink if it has a severity 1 error.
- Yellow if it has a severity 2 error.
- White if it has a severity 3 error.

Selected log entries have a background color of:

- Red if it has a severity 1 error.
- Green if it has a severity 2 error.
- Blue if it has a severity 3 error.

These colors are configurable and can be changed in the Log Analyzer's Preferences Log page. Select **File -> Preferences -> Logs -> Severity**.

The Log Analyzer can also display the log entries in different sorting sequences. Select **File -> Preferences -> Logs**.

After the analyze action has been invoked, each analyzed log entry has the following icons:

✓ The tick icon indicates that the entry has some analysis information in one or more pages in the analysis pane.

✚ The plus icon indicates that the entry has some analysis information. Look at the log entry prior to this one when diagnosing problems.

? The question mark icon indicates that the entry has either a severity 1 or 2 error but no additional analysis information is available for it.

✗ The cross icon indicates that the entry has a severity 3 error and it has no analysis information.

Record pane (upper right)

When you select an entry under the unit of work in the logs pane, you see the details of the entry (process ID, thread ID, server name, severity, etc.) in the Record pane. The entry's identification is shown in the pane's title bar. Right-click in this record pane to see the actions that you can perform on the entry (Analyze, Save to File, Find, Select All).

There is a drop-down arrow to the left of “Record” in the pane’s title bar, which allows you to go back to look at the last 10 records that you have viewed. The default cache size for the historical data is 10. Select **File -> Preferences -> General** to modify this.

Analysis pane (lower right)

When the analyze action has been invoked, any information found in the symptoms database for the selected log entry will appear in the symptom page. If the page tab is grayed out, there is no information in that page.

There is a status line at the bottom of the window showing the status of actions.

15.5.2 Merging logs on multiple application servers

The correlation ID can be used to correlate activity to a particular client request, or correlate activities on multiple application servers.

To merge different service or activity.log files from different machines where your transaction occurred:

1. Make sure that Enable Correlation ID box is checked as discussed in 15.3.3, “IBM service (activity) log” on page 725.
2. Open one of the files in Log Analyzer and select **File -> Preferences -> Logs** to sort the log records in UnitOfWork and TimeStamp order to have a distributed log view.

3. Use the **File -> Merge with** option to merge files.

15.5.3 Updating the symptom database

The symptom database included in the Log Analyzer package contains entries for common events and errors. New versions of the symptom database provide additional entries.

You can download the symptoms database using the Log Analyzer GUI. Select **File -> Update Database -> WebSphere Application Server Symptom Database** (for WebSphere Application Server) or **WebSphere Application Server Network Deployment Symptom Database** (for WebSphere Application Server Network Deployment) from the main menu, as shown in Figure 15-13.

Alternatively, you can download new versions of the database from the IBM FTP site. The URL for the FTP site is located in the <WAS_HOME>/properties/logbr/ivblogbr.properties file or the <WAS_ND_HOME>/properties/logbr/ivblogbr.properties file.

The symptom files are located in the <WAS_HOME>/properties/logbr/symptoms file or the <WAS_ND_HOME>/properties/logbr/symptoms file.

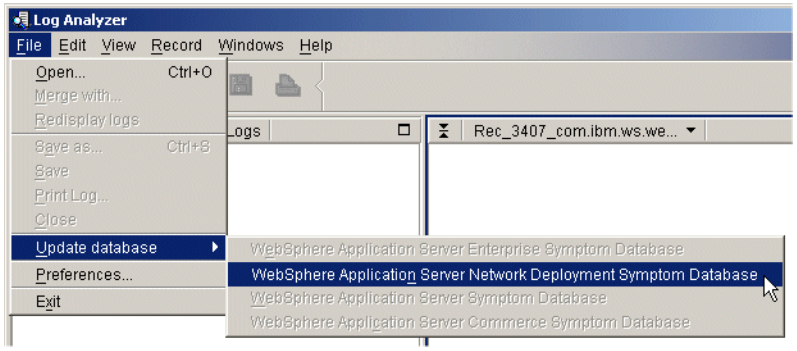

Log Analyzer
File
Edit
View
Record
Windows
Help
Open... Ctrl+O
Merge with...
Redisplay logs
Save as... Ctrl+S
Save
Print Log...
Close
Update database
Preferences...
Exit
Logs
Rec_3407_com.ibm.ws.we...
WebSphere Application Server Enterprise Symptom Database
WebSphere Application Server Network Deployment Symptom Database
WebSphere Application Server Symptom Database
WebSphere Application Server Commerce Symptom Database

Figure 15-13 Updating the symptom database

If your organization uses an FTP or SOCKS proxy server, you can add a proxy definition to the Proxy Preferences page as described below:

1. Select **File -> Preferences -> Proxy**.
2. Select the appropriate proxy type.
3. Enter the host name and port number of the proxy server on the Proxy window.

15.6 Thread Analyzer technology preview

In the past we have had the ability to determine the CPU or memory usage involved in processing applications but lacked the ability to look deeper into the application process. JVM problem determination, especially in the multi-threaded WebSphere Application Server environment, has been difficult due to insufficient analysis tools. When performing problem determination, it is often necessary to know exactly what is happening in the application server or in the application code.

In addition, application servers sometimes produce a Java stack trace, also called a Java thread dump or javacore file, as a result of errors. Tools were needed to assist in viewing this trace.

The Thread Analyzer technical preview can assist you in analyzing Java stack trace files and in viewing the threads, both for hung or deadlock conditions, or to simply evaluate application processing. However, Thread Analyzer is simply a tool to help you view threads and stack traces. The analysis is still up to you, so we will first take a look at the Java stack trace structure.

You can download the Thread Analyzer at:

`http://www7b.boulder.ibm.com/wsdd/downloads/ta.html`

15.6.1 Understanding the Java stack trace

When a JVM crashes with an internal error, for example a segmentation violation (`SIGSEGV`, signal # 11) , an illegal instruction (`SIGILL`, signal # 9) or an abort signal (`SIGABRT`, signal #6), it will call its own signal handler to dump thread and monitor information.

If you are using the IBM JDK, the Java stack trace will be dumped to the special javacore file in addition to the system core dump file. The javacore file name will be in the following format:

javacore.<process_id>.<time>.txt

where <time> is the return value from the time subroutine at the time of the `core` dump.

The javacore file will be located in one of the following locations, in this order:

1. The directory referred to by the environment variable IBM_JAVACOREDIR.
2. The current working directory of the JVM process.
3. The directory defined by the TMPDIR environment variable .
4. The “/tmp” directory.

Normally, on AIX and Windows, the directory is <WAS_HOME> directory (not <WAS_HOME>/bin as in previous versions). Sometimes, the Java stack trace will be generated in the <stderr> log file or on your foreground terminal console if your platform is Solaris or HP-UX.

Generating the Java stack trace

The Java stack trace or thread dump can be generated explicitly in one of two ways:

- For UNIX platforms: Send a specific signal explicitly to the JVM process, for example signal SIGQUIT (signal #3) or SIGILL (signal #4), SIGTERM (signal # 15), as follows:

  ```
  # kill -3 <unix_process_id>
  ```

  In the case of the SIGQUIT signal, the WebSphere process does not stop running because it has its own signal handler. The JVM will generate the Java stack trace and keep going.

  Note: Do not send signal #3 SIGQUIT frequently to the process on heavily loaded systems because it could be temporarily blocked or, on rare occasions, crash your application server.

  Note: Do not use the command **`kill -9 <pid>`** which means sending a signal #9 SIGKILL. This command kills the process without trace results.

  The signal numbers and their meanings are listed in Table C-1 on page 914.

- Using debugging tools or API calls, for instance the Thread Analyzer.

If the Java stack trace file generation is successful, the file will look as follows:

Example 15-9 Java stack trace

```
Fri Nov  8 10:41:53 2002
SIGQUIT received at 0x0 in <unknown>.
J2RE 1.3.1 IBM AIX build ca131-20020821
Current Thread Details
----------------------
    "Signal dispatcher" sys_thread_t:0x40BD2A28
         ----- Native Stack -----
        unavailable - iar 0x0 not in text area
------------------------------------------------------------------------
Operating Environment
---------------------
Host                    : m10df51f.itso.ral.ibm.com:9.24.104.21
OS Level                : AIX 5.1.0.0
```

```
Processors -
        Architecture    : POWER_PC (impl: POWER_630, ver: PV_630)
        How Many        : 1
        Enabled         : 1
User Limits (in bytes except for NOFILE and NPROC) -
        RLIMIT_FSIZE    : 1073741312
        RLIMIT_DATA     : 2147483645
        RLIMIT_STACK    : 33554432
        RLIMIT_CORE     : 1073741312
        RLIMIT_NOFILE   : 32000
        NPROC(max)      : 262144
Page Space (in blocks) -
        /dev/hd6: size=131072, free=129917
Application Environment
-----------------------
Signal Handlers -
        SIGHUP          : intrDispatchMD (libhpi.a)
        ....
Environment Variables -
_=/usr/WebSphere/AppServer/java/bin/java
LANG=en_US
CONFIG_ROOT=/usr/WebSphere/AppServer/config
USER=root
WAS_HOME=/usr/WebSphere/AppServer
WAS_CELL=m10df51fNetwork
WAS_NODE=m10df51f
...
Loaded Libraries (sizes in bytes)
---------------------------------
/usr/WebSphere/AppServer/java/jre/bin/liborb.a
        filesize        : 9738
        text start      : 0xD32B9000
        text size       : 0x1849
        data start      : 0x43E4FE28
        data size       : 0x118
...
---------------------- Exception Information ---------------------------
No Exception

---------------------- System Properties ------------------------------
J2RE 1.3.1 IBM AIX build ca131-20020821
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (J2RE 1.3.1 IBM AIX build ca131-20020821, native
threads):
    "Servlet.Engine.Transports : 3" (TID:0x32E47D10, sys_thread_t:0x44831928,
state:CW, native ID:0x
2532) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ....
        at com.ibm.ws.webcontainer.http.HttpConnection.handleRequest(HttpCo...)
        at com.ibm.ws.http.HttpConnection.readAndHandleRequest(HttpConnecti...)
        at com.ibm.ws.http.HttpConnection.run(HttpConnection.java(Compiled...))
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:546)
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F3A430 in lkMonitorEnter
        at 0xD3078154 in _jit_monitorEnterQuicker
        at 0xD307DB30 in JITSigSegvHandler
...
---------------------- LK component Dump Routine  ----------------------
Monitor pool info:
...
Monitor Pool Dump (flat & inflated object-monitors):
...
JVM System Monitor Dump (registered monitors):
...
Thread identifiers (as used in flat monitors):
...
Java Object Monitor Dump (flat & inflated object-monitors):
...
---------------------- END OF DUMP -------------------------------------
```

The javacore file can be grouped into three distinct sections:

- **Runtime environments**: current thread details, operating environment, application environment, loaded libraries, exception information, and system properties.
- **Thread dumps**: full thread dump.
- **Monitor information**: monitor pool info, monitor pool dump, JVM system monitor dump, thread IDs, Java object monitor dump.

Understanding thread status

When diagnosing a Java stack trace you will need to understand thread status. Example 15-10 on page 748 shows a following thread entry with a state of "CW".

Example 15-10   Thread entry example

```
"Servlet.Engine.Transports : 3" (TID:0x32E47D10, sys_thread_t:0x44831928,
state:CW, native ID:0x
2532) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ....
```

The following are the thread status indicators found in a IBM JDK stack trace:

- **R**: Running or runnable thread.
- **CW**: Condition Wait. The thread is waiting on a condition variable. For example, Thread.sleep() and Object.wait() wait in CW state until notified. The CW state is sometimes referred to as “waiting to be notified” or “waiting on condition” in the Thread Full Dump section of the dump file.
- **MW:** Monitor Wait. The thread is waiting on a monitor lock. If more than one thread tries to enter a synchronized block concurrently, only one thread can enter. Others have to wait until the first thread releases the monitor lock. In this case, the status of the other threads is MW. The MW status is sometimes referred to as “waiting to enter”, “waiting for monitor entry”, or “waiting to lock monitor” in the Thread Full Dump section of the dump file.
- **S:** Suspended thread
- **MS:** Monitor Suspended. The thread suspended waiting on a monitor lock.

Monitors are data structures used for synchronization within the JVM. Each object has a monitor associated with it. A monitor can be thought of as a lock of an object. If another thread already owns the monitor associated with the lock object, the current thread waits until the object is unlocked, then tries again to gain ownership.

In normal cases, you will see threads in R and CW status, and sometimes MW.

If you see a thread in MS status, you can suspect a JVM problem because a thread in MW status will often appear in the S status when it is suspended.

Case 1: Thread status MW (Monitor Wait)

Take a look at the code in Example 15-11 on page 749.

Example 15-11 MWtest.jsp for threads in status R and MW

```
<%@ page session="false" contentType="text/html" %>
<%! public static Object lock = new Object(); %>
<html><body>MWtest.jsp for threads in status R and MW
<%
  synchronized(lock){
    for (long i=0;i<500000000L;i++);
  }
%>
</body></html>
```

In a situation where the JSP is requested continuously and concurrently from multiple clients, the state of the JVM stack could be as shown in Example 15-12.

Example 15-12 Java stack trace result of MWtest.jsp

```
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (....):
    ......
    "Servlet.Engine.Transports : 6" (TID:0x131872B8, sys_thread_t:0x2418A9B8,
state:MW, native ID:0x844) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
    "Servlet.Engine.Transports : 4" (TID:0x13C15190, sys_thread_t:0x241876E0,
state:MW, native ID:0x9D8) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
    "Servlet.Engine.Transports : 3" (TID:0x14002138, sys_thread_t:0x2416F2D8,
state:R, native ID:0x920) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
    "Servlet.Engine.Transports : 0" (TID:0x137ACA68, sys_thread_t:0x23FC9718,
state:MW, native ID:0x948) prio=5
   at org.apache.jsp._MWtest._jspService(_MWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
    ...
---------------------- LK component Dump Routine  ----------------------
Monitor pool info:
  ...
Monitor Pool Dump (flat & inflated object-monitors):
  ...
  sys_mon_t:0x007B9408 infl_mon_t: 0x007B8F58:
    java.lang.Object@13186888/13186890: owner "Servlet.Engine.Transports : 3" 
(0x2416F2D8), entry count 1
   Waiting to enter:
       "Servlet.Engine.Transports : 6" (0x2418A9B8)
       "Servlet.Engine.Transports : 4" (0x241876E0)
       "Servlet.Engine.Transports : 0" (0x23FC9718)
  ...
Java Object Monitor Dump (flat & inflated object-monitors):
    ...
    java.lang.Object@13186888/13186890
        locknflags 80001600 Monitor inflated infl_mon 0x007B8F58
---------------------- END OF DUMP ------------------------------------
```

You can see that only one thread, Servlet.Engine.Transports:3, is running in thread state R. This is because only one thread can enter the synchronized block associated with the lock object. The others are waiting to enter the block and so appear as having thread status MW (waiting on the monitor lock). In the Monitor Pool Dump section at the bottom, the owner of the monitor lock, java.lang.Object@13186888/13186890, is Servlet.Engine.Transports:3, and others are listed as the “Waiting to enter” (thread state MW).

If there is an abnormally large number of threads in MW state, it could indicate that your application has a performance problem by design. Usage of synchronized code blocks should be used very carefully.

Case 2: Thread status CW (Condition Wait)

The code shown in Example 15-13 is an example of the CW thread status (waiting on a condition variable).

Example 15-13 CWtest.jsp for threads in status R and CW

```
<%@ page session="false" contentType="text/html" %>
<%!
  public static Object lock = new Object();
  public static final int MAX_RUNNABLE_THREADS = 2;
  public static int running_thread_count = 0;
%>
<html><body>CWtest.jsp for threads in status R and CW
<%
  try{
    synchronized(lock){
      while( running_thread_count >= MAX_RUNNABLE_THREADS )
        try{ lock.wait(1000); }catch(Exception e){}
      ++running_thread_count;
    }
    //-------------------------------------------------
    for (long i=0;i<500000000L;i++);
    //-------------------------------------------------
  }
  finally{
    synchronized(lock){
      --running_thread_count;
      lock.notifyAll();
    }
  }
%>
</body></html>
```

This code is similar to the previous example, but we use the wait() and notify/All() thread APIs to control the maximum number of running threads at the same time.

The Java stack trace looks like this:

Example 15-14 Java stack trace result of CWtest.jsp

```
...
---------------------- XM component Dump Routine  ----------------------
Full thread dump Classic VM (....):
    "Servlet.Engine.Transports : 9" (TID:0x13329B60, sys_thread_t:0x2415EB78,
state:CW, native ID:0xAB8) prio=5
   at java.lang.Object.wait(Native Method)
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 8" (TID:0x13329BC0, sys_thread_t:0x24129B30,
state:R, native ID:0xA40) prio=5
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 2" (TID:0x13019B20, sys_thread_t:0x23FF8CC8,
state:CW, native ID:0x6A8) prio=5
   at java.lang.Object.wait(Native Method)
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
   ...
    "Servlet.Engine.Transports : 1" (TID:0x13019EC8, sys_thread_t:0x2400EFC8,
state:CW, native ID:0x700) prio=5
   at java.lang.Object.wait(Native Method)
   at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
   at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
   at javax.servlet.http.HttpServlet.service(...)
```

```
    ...
     "Servlet.Engine.Transports : 0" (TID:0x10F043F0, sys_thread_t:0x23CDF5B8,
state:R, native ID:0x954) prio=5
    at org.apache.jsp._CWtest._jspService(_CWtest.java(Compiled Code))
    at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(...)
    at javax.servlet.http.HttpServlet.service(...)
    ...
---------------------- LK component Dump Routine  ----------------------
    ...
Monitor Pool Dump (flat & inflated object-monitors):
    ...
  sys_mon_t:0x007B93E0 infl_mon_t: 0x007B8F38:
    java.lang.Object@132A9D10/132A9D18: <unowned>
    Waiting to be notified:
        "Servlet.Engine.Transports : 9" (0x2415EB78)
        "Servlet.Engine.Transports : 1" (0x2400EFC8)
        "Servlet.Engine.Transports : 2" (0x23FF8CC8)
Java Object Monitor Dump (flat & inflated object-monitors):
    ...
    java.lang.Object@132A9D10/132A9D18
        locknflags 80001500 Monitor inflated infl_mon 0x007B8F38
...
```

Only two threads, Servlet.Engine.Transports:0 and 8, are running in state R because the total number of runnable threads was restricted by the MAX_RUNNABLE_THREADS variable. The others, threads #1,2, and 9, are waiting to be notified in CW state at the Object.wait(). This information is also listed in the Monitor Pool Dump under the “Waiting to be notified” line for the monitor lock object, java.lang.Object@132A9D10/132A9D18.

When the threads in CW state are notified of a notify event, only one thread can access that object exclusively. Even when the previous thread has sent a notify event to the waiting threads, the waiting threads can’t access the synchronized block until the notifying thread has left its synchronized block.

Be careful when you analyze threads in CW state because this can be a normal situation. All threads in a thread pool (for example, the worker-threads named Servlet.Engine.Transports of the Web container) are always waiting in CW state to be notified of new requests, as shown in Example 15-15:

Example 15-15 WebSphere V5 Servlet engine thread is waiting for new request

```
"Servlet.Engine.Transports : 0" (TID:0x32246558, sys_thread_t:0x4482B9F8,
state:CW, native ID:0x1E2F) prio=5
        at java.lang.Object.wait(Native Method)
        at java.lang.Object.wait(Object.java(Compiled Code))
        at com.ibm.ws.util.BoundedBuffer.take(BoundedBuffer.java(CompiledCode))
        at com.ibm.ws.util.ThreadPool.getTask(ThreadPool.java(Compiled Code))
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:553)
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F39728 in lkMonitorWait
        at 0xD2EAB150 in JVM_MonitorWait
```

On the other hand, let's take a case involving the JDBC connection pool. If there are lots of threads in state CW waiting to be notified of a free connection reference, this implies that all connections are held by other threads or some connections were not properly returned by the application to the connection pool. This is a case to pursue.

It is not easy to know which thread has to send a notify event to the waiting threads. If the thread completed but did not send a notify, the current thread is not the one that holds the lock.

This means that there is no direct relation between the current (suspected) thread and the waiting threads. We can only assume that the actual thread is one of any of the running/runnable threads and even this might not be the case if the thread has entered a CW or MW state caused by another, unrelated monitor lock. The analysis process depends purely on your intuition, experience, and familiarity with the code.

Case 3: deadlock

This last example is to illustrate deadlocked threads. If you call the JSP shown in Example 15-16 once there is no problem. But if you have multiple requests you will not get a response.

Example 15-16 Deadlocked threads

```
<%@ page session="false" contentType="text/html" %>
<%! public static Object monitor1 = new Object();
    public static Object monitor2 = new Object();
    public static boolean toggle = true;
%>
<html><body> Deadlock threads test
<%
  Object lock1 = (toggle)? monitor1:monitor2;
  toggle = !toggle;
  synchronized(lock1){
    for (long i=0;i<200000000L;i++);

    Object lock2 = (lock1==monitor1)? monitor2:monitor1;
    synchronized(lock2){
      for (long i=0;i<200000000L;i++);
    }
  }
%>
</body></html>
```

Look at the following Java stack trace shown in Example 15-17.

Example 15-17 Java stack trace of deadlocked threads

```
Full thread dump Classic VM (J2RE 1.3.1 IBM AIX build ca131-20020821, native
threads):
------------------------------------------------------------------------
    "Servlet.Engine.Transports : 3" (TID:0x32E47D10, sys_thread_t:0x44831928,
state:CW, native ID:0x
2532) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ...
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F3A430 in lkMonitorEnter
        at 0xD3078154 in _jit_monitorEnterQuicker
        at 0xD307DB30 in JITSigSegvHandler
------------------------------------------------------------------------
    "Servlet.Engine.Transports : 2" (TID:0x32E47D70, sys_thread_t:0x44831518,
state:CW, native ID:0x
2431) prio=5
        at org.apache.jsp._deadlock._jspService(_deadlock.java(Compiled Code))
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ...
         ----- Native Stack -----
        at 0xD00549DC in _event_sleep
        at 0xD0054ECC in _event_wait
        at 0xD0060E08 in _cond_wait_local
        at 0xD006129C in _cond_wait
        at 0xD0061FC0 in pthread_cond_wait
        at 0xD3056164 in condvarWait
        at 0xD3055120 in sysMonitorWait
        at 0xD2F3A430 in lkMonitorEnter
        at 0xD3078154 in _jit_monitorEnterQuicker
        at 0xD307DB30 in JITSigSegvHandler
---------------------- LK component Dump Routine  ----------------------
Monitor Pool Dump (flat & inflated object-monitors):
  sys_mon_t:0x3020F558 infl_mon_t: 0x00000000:
    java.lang.Object@328781F0/328781F8: Flat locked by thread ident 0x27, entry
count 1
        Waiting to be notified:
            "Servlet.Engine.Transports : 2" (0x44831518)
  sys_mon_t:0x3020F5D8 infl_mon_t: 0x00000000:
    java.lang.Object@32878200/32878208: Flat locked by thread ident 0x26, entry
count 1
        Waiting to be notified:
            "Servlet.Engine.Transports : 3" (0x44831928)
  ...
Thread identifiers (as used in flat monitors):
    ident 0x27 "Servlet.Engine.Transports : 3" (0x44831928) ee 0x4483171C
    ident 0x26 "Servlet.Engine.Transports : 2" (0x44831518) ee 0x4483130C
  ...
```

First, there are two servlet worker threads, Servlet.Engine.Transports : 3 and 2 in CW state. You can see that the identifiers of the threads are 0x27 and 0x26 respectively in the Thread identifiers section.

In the Monitor Pool Dump section we can see that the monitor lock objects that are locked by these threads are Object@328781F0/328781F8 and Object@32878200/32878208 respectively.

This is easier to look at if we put it in a table.

Table 15-2 Threads relation table

thread name	thread ident	monitor locked by the thread	thread waiting to be notified
`Servlet.Engine.Transports : 3`	**0x27**	`Object@328781F0/328781F8`	`Servlet.Engine.Transports : 2`
`Servlet.Engine.Transports : 2`	**0x26**	`Object@32878200/32878208`	`Servlet.Engine.Transports : 3`

The key is in the last column. The two threads each have their own lock monitors and are waiting to be notified by the other monitor. The two threads are at total deadlock.

Flat and Inflated monitors

It is normal for synchronized accesses to occur since only one thread at a time can access the object or region of code. Therefore, the overhead of setting up a monitor for the first thread entering the synchronized block is unnecessary. The first "locking thread" simply sets a flag on the object to say that the object is locked (by that thread) and proceeds onwards - this is a "flat" monitor. If a second thread needs to enter the synchronized block, then the full monitor operation takes place - the monitor is "inflated". We have thus postponed (and often avoided) the cost of setting up the monitor.

Therefore, a flat monitor implies that only one thread is accessing the object or region of code. An inflated monitor implies that multiple threads are trying to gain access to the monitor.

Common system monitors

There are several JVM system monitors. Understanding the system monitors sometimes helps us to be able to understand the overall thread status in the Java virtual machine. The following table lists the JVM common registered monitors.

Table 15-3 JVM common registered monitors

Name	Description
Evacuation Region lock	
Heap Promotion lock	
Integer lock access-lock	
Sleep lock	
Method trace lock	
UTF8 Cache lock	
Heap lock	Protects the Java heap during heap memory.
Rewrite Code lock	Protects code when an optimization is attempted.
Monitor Cache lock	Only one thread can have access to the monitor cache at a time. This lock ensures the integrity of the monitor cache.
JNI Pinning lock	Protects block copies of arrays to native method code
JNI Global Reference lock	Locks the global reference table that holds values that need to be explicitly freed, and will outlive the lifetime of the native method call.
Class loader/loading lock	Ensures only one thread loads a class at a time.

Name	Description
Binclass lock	Locks access to the loaded and resolved classes.
Class linking lock	Protects a class's data when loading native libraries to resolve symbolic references.
Monitor Registry lock	Only one thread can have access to the monitor registry at a time. This lock ensures the integrity of that registry.
Thread queue lock	Protects the queue of active threads.
Verifier lock	
Name and type hash table lock	Protects the JVM hash tables of constants and their types.
String intern lock	Locks the hashtable of defined strings that were loaded from the class constant pool.
Zip lock	
Java stack lock	Protects the free stack segments list.
Has finalization queue lock	Protects the lists of queue lock objects that have been garbage-collected, and deemed to need finalization. They are copied to the Finalize me queue.
Finalize me queue lock	Protects a list of objects that can be finalized at leisure.

15.6.2 Running Thread Analyzer

It might be hard for you to read the text-based Java stack trace file, which often contains hundreds or thousands of lines. Thread Analyzer will assist you in reading and analyzing the trace.

Starting Thread Analyzer

To run Thread Analyzer, you will need to set the WAS_HOME variable to the install path for WebSphere Application Server. Example 15-18 shows how to start Thread Analyzer.

Example 15-18 Starting Thread Analyzer

```
C:\ThreadAnalyzer\bin>set WAS_HOME=c:\WebSphere\AppServer
C:\ThreadAnalyzer\bin>tagui.cmd
```

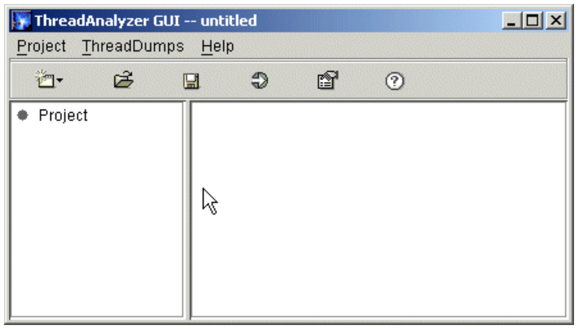


Figure 15-14 Starting Thread Analyzer

Obtaining a thread dump to analyze

There are several ways to get a Java stack trace (or thread dump) into the Thread Analyzer.

From an existing Java stack trace file

1. Select **ThreadDumps -> Open Existing File**.
2. Select and open a Java stack trace file (normally named javacore.???.txt) to analyze.

From a server output log file

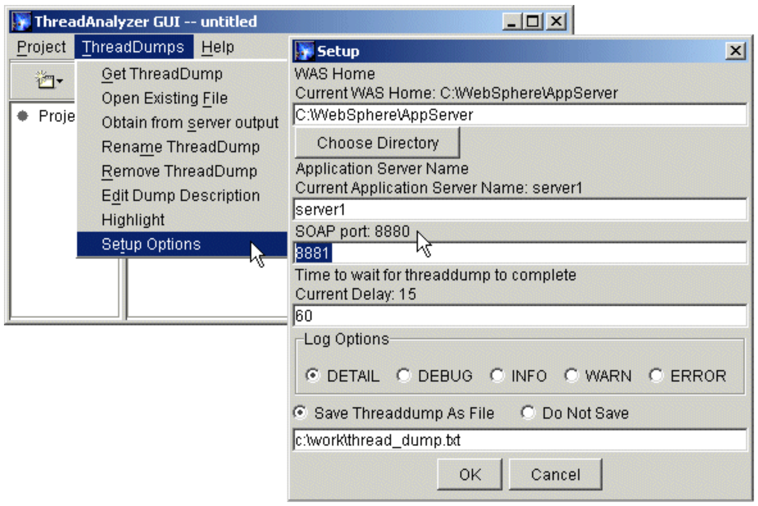
You can take a Java stack trace source from application server's log files, <stderr> or <stdout> files or any other snippets of files that contain a thread dump.

1. Select **ThreadDumps -> Obtain from server output**.
2. Click **Add files** on the new pop-up window to select files one or more times.
3. Click **Process** button to extract the only Java stack trace parts from the files.

Getting a thread dump in real time

You can also generate a real-time thread dump while the application server is alive.

1. Before you get a thread dump from an application server alive, you should set up the options by selecting **ThreadDumps -> Setup Options**, as shown in Figure 15-15 on page 759.



WAS Home
Current WAS Home: C:\WebSphere\AppServer
C:\WebSphere\AppServer

Application Server Name
Current Application Server Name: server1
server1

SOAP port: 8880
8881

Time to wait for threaddump to complete
Current Delay: 15
60

Log Options
[✓ DETAIL] DEBUG INFO WARN ERROR

[✓ Save Threaddump As File] Do Not Save
c:\work\thread_dump.txt

Figure 15-15 Thread Analyzer - Setup Options

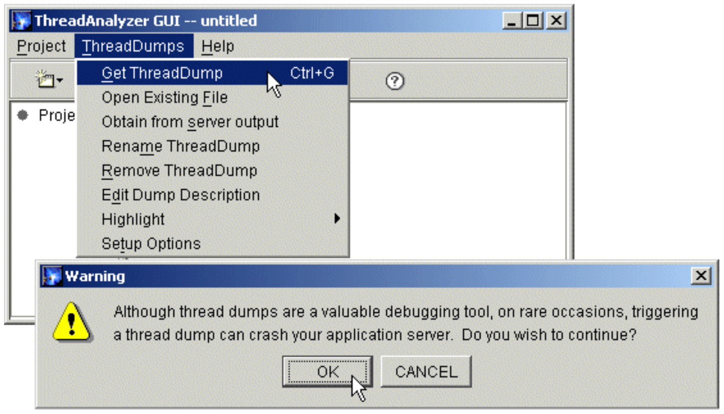
a. In the pop-up window, choose the WebSphere Application Server installation directory.

b. Type in the name of the application server you want to analyze.

c. Verify or select the SOAP port number. You should first check this port number from the WebSphere administrative console by selecting **Servers -> Application Servers** in the console navigation tree. Open the server configuration and select **End Points** in the Additional Properties table. The port number will be in the SOAP_CONNECTOR_ADDRESS configuration.

d. Change or take the default for the wait time, log options and stack trace saving options.

2. Now, you can generate a thread dump from a running application server by clicking **ThreadDumps -> Get ThreadDump,** or by simply pressing Ctrl+G as shown in Figure 15-16 on page 760.

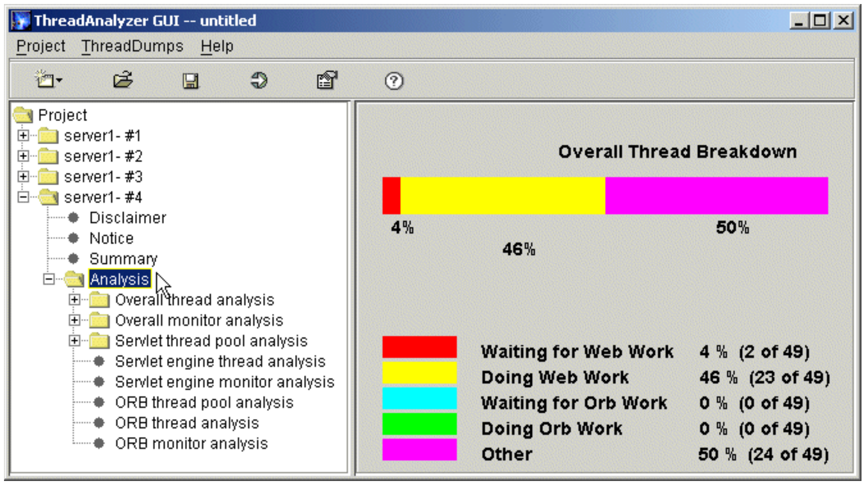


Although thread dumps are a valuable debugging tool, on rare occasions, triggering a thread dump can crash your application server. Do you wish to continue?

Figure 15-16 Thread Analyzer - Getting ThreadDump

Navigating Thread Analyzer

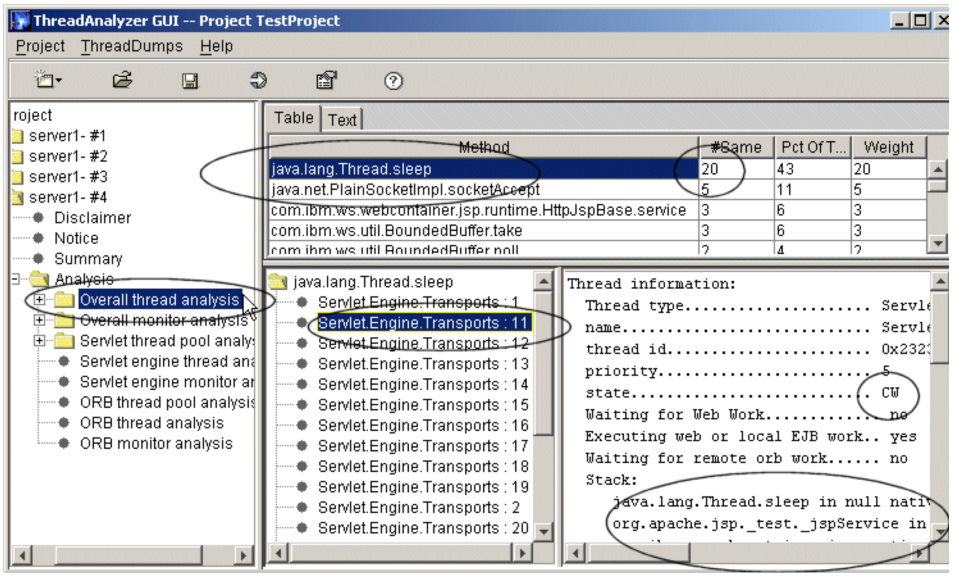
If you are successful at generating or loading the Java stack trace or thread dump you should see something like in Figure 15-17.



🟥	Waiting for Web Work	4 % (2 of 49)
🟨	Doing Web Work	46 % (23 of 49)
🟦	Waiting for Orb Work	0 % (0 of 49)
🟩	Doing Orb Work	0 % (0 of 49)
🟪	Other	50 % (24 of 49)

Figure 15-17 Thread Analyzer - Result of ThreadDump

You can see that 23 servlet engine worker threads are doing something now and another two threads are waiting for a new request. For more detailed information about what the threads are doing, you can use the overall thread analysis for all threads state or servlet thread pool analysis for servlet threads in the left navigation tree, as shown in Figure 15-18 on page 761.


ThreadAnalyzer GUI -- Project TestProject
Project
ThreadDumps
Help
roject
server1- #1
server1- #2
server1- #3
server1- #4
Disclaimer
Notice
Summary
Analysis
Overall thread analysis
Overall monitor analysis
Servlet thread pool analy
Servlet engine thread an
Servlet engine monitor ar
ORB thread pool analysis
ORB thread analysis
ORB monitor analysis
Table
Text
java.lang.Thread.sleep
Servlet.Engine.Transports : 1
Servlet.Engine.Transports : 11
Servlet.Engine.Transports : 12
Servlet.Engine.Transports : 13
Servlet.Engine.Transports : 14
Servlet.Engine.Transports : 15
Servlet.Engine.Transports : 16
Servlet.Engine.Transports : 17
Servlet.Engine.Transports : 18
Servlet.Engine.Transports : 19
Servlet.Engine.Transports : 2
Servlet.Engine.Transports : 20

Method	#Same	Pct Of T...	Weight
java.lang.Thread.sleep	20	43	20
java.net.PlainSocketImpl.socketAccept	5	11	5
com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service	3	6	3
com.ibm.ws.util.BoundedBuffer.take	3	6	3
com.ibm.ws.util.BoundedBuffer.poll	2	4	2

```
Thread information:
  Thread type..................... Servle
  name............................ Servle
  thread id....................... 0x232
  priority........................ 5
  state........................... CW
  Waiting for Web Work............ no
  Executing web or local EJB work.. yes
  Waiting for remote orb work...... no
  Stack:
    java.lang.Thread.sleep in null nati
    org.apache.jsp._test._jspService in
```

Figure 15-18 Thread Analyzer Overall thread analysis

15.7 Collector tool

The Collector tool gathers information about a WebSphere Application Server installation and packages it in an output JAR file. The file can be sent to IBM Customer Support to assist in problem determination and analysis. The information in the file includes logs, property files, configuration files, operating system and Java data, and prerequisite software presence and levels.

The -Summary option, available with V5.0.2, is useful for determining the features installed. It produces a text file.

Running the Collector tool

The Collector tool should be run under the root or administrator user ID because some of the commands to be executed require system access authority. However, if this isn't possible and you proceed without administrator authority, much of what the Collector does will work just fine.

The Collector tool writes its output files to the current directory, so it is a good idea to create a new directory from which to run the tool. You cannot run the Collector tool in a directory under the WebSphere Application Server installation directory.

- For Windows systems, log on to the system as administrator or another user with administrator authority and enter the following:

  ```
  C:\> mkdir work
  C:\> cd work
  C:\work> c:\WebSphere\AppServer\bin\collector.bat
  ```

- For UNIX systems, log on to the system as root and do the following:

  ```
  itsosvr:/home/# id
  root(....)
  itsosvr:/home/# mkdir work
  itsosvr:/home/# cd work
  itsosvr:/home/work# /usr/WebSphere/AppServer/bin/collector.sh
  ```

To collect information in a Network Deployment environment, invoke the Collector tool from the Deployment Manager install directory instead.

Results

The Collector program creates a log file, Collector.log, and an output .jar file in the current work directory. The .jar file name is based on the host name and package of the server on which the Collector tool was run, in the format: hostname-ND|Base-WASenv.jar.

What to do next

Send the hostname-ND|Base-WASenv.jar file to IBM Customer Support for analysis.

15.8 First Failure Data Capture logs

The First Failure Data Capture (FFDC) function preserves the information generated from a processing failure and returns control to the affected engines. There are three property files which control the behavior of the FFDC filter:

- properties/ffdcStart.properties - used while the server is starting
- properties/ffdcRun.properties - used after the server is ready
- properties/ffdcStop.properties - used while the server is stopping

The captured data is saved automatically in the <WAS_HOME>/logs/ffdc directory for use in analyzing the problem, and could be collected by the Collector tool.

The First Failure Data Capture tool is intended primarily for use by IBM Service. It runs as part of the WebSphere Application Server and you cannot start or stop it. It is recommended that you not attempt to configure the FFDC tool. If you experience conditions requiring you to contact IBM Service, your IBM Service representative will assist you in reading and analyzing the FFDC log.

15.9 Dumping the contents of the name space

The name space stored by a given name server can be dumped with the dumpNameSpace utility that is shipped with WebSphere Application Server. This utility can be invoked from the command line or from a Java program. The naming service for the WebSphere Application Server host must be active when this utility is invoked.

To invoke the utility through the command line, enter the following command from the <WAS_HOME>/bin directory:

- UNIX:

  ```
  dumpNameSpace.sh [[-keyword value ]...]
  ```

- Windows:

  ```
  dumpNameSpace [[-keyword value ]...]
  ```

The following command shows how to invoke the dumpNameSpace utility from the command line:

```
dumpNameSpace -?
```

The generated output will look like Example 24-6, which is the *short* dump format.

```
dumpNameSpace -host localhost -report short
```

Example 15-19 dumpNameSpace output

```
Getting the initial context
Getting the starting context
=============================================================================
Name Space Dump
   Provider URL: corbaloc:iiop:localhost:2809
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: cell
   Starting context: (top)=ItsoNetwork
   Formatting rules: jndi
   Time of dump: Tue Oct 22 15:57:37 EDT 2002
=============================================================================

=============================================================================
Beginning of Name Space Dump
=============================================================================
```

```
    1 (top)
    2 (top)/cell                                      javax.naming.Context
    2    Linked to context: ItsoNetwork
    3 (top)/legacyRoot                                javax.naming.Context
    3    Linked to context: ItsoNetwork/persistent
    4 (top)/cells                                     javax.naming.Context
    5 (top)/deploymentManager                         javax.naming.Context
    5    Linked to URL: corbaloc::ka0klfr:9809/NameServiceServerRoot
    6 (top)/persistent                                javax.naming.Context
    7 (top)/persistent/cell                           javax.naming.Context
    7    Linked to context: ItsoNetwork
...
   30 (top)/nodes/ka0klfrManager/node                 javax.naming.Context
   30    Linked to context: ItsoNetwork/nodes/ka0klfrManager
   31 (top)/cellname                                  java.lang.String
   32 (top)/clusters                                  javax.naming.Context
==============================================================================
End of Name Space Dump
==============================================================================
```

15.10 HTTP session monitoring

In the event of session-related problems, it is helpful to collect all session-related information. WebSphere Application Server V5 introduces an HTTP session tracker servlet called IBMTrackerDebug. To access the servlet from a browser, use the following URL:

```
http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDeb
ug
```

Example 15-20   Result of IBMTrackerDebug servlet

```
J2EE NAME(AppName#WebModuleName):: DefaultApplication#DefaultWebApplication.war
cloneId : -1

Number of sessions in memory: (for this webapp) : 11
use overflow : true
overflow size (for this webapp) :
Invalidation alarm poll interval (for this webapp) : 304
Max invalidation timeout (for this webapp) : 1800
Using Cookies : true
Using URL Rewriting : false
use SSLId : false
URL Protocol Switch Rewriting : false
Session Cookie Name : JSESSIONID
Session Cookie Comment : SessionManagement
Session Cookie Domain : null
Session Cookie Path : /
Session Cookie MaxAge : -1
Session Cookie Secure : false
Maximum in memory table size : 1000
current time : Wed Oct 23 18:18:37 EDT 2002
integrateWASSec :false
Session locking : false
Session locking timeout: 5
Allow access on lock timeout:true
Sessions Created:11
Active Count:0
Session Access Count:8
Invalidated Sessions Count:0
Invalidated By SessionManager:0
Garbage Collected count:0
SessionAffinity Breaks:0
Number of times invalidation alarm has run:0
Rejected Session creation requests(overflow off):0
Cache Discards:0
Attempts to access non-existent sessions:2
Number of binary reads from external store:0
Total time spent in reading from external store(ms):0
Total number of bytes read:0
Number of binary writes to external store:0
Total time spent in writing to external store(ms):0
Total number of bytes wriiten out:0
Total size of serializable objects in memory :1859
Total number objects in memory :11
Min size session object size:169
Max size session object size :169
```

If the IBMTrackerDebug servlet does not give you enough information to solve an HTTP session-related problem, you can create your own monitoring servlet to view or dump all the HTTP session data in memory.

For more detailed information, see Appendix C, “Additional troubleshooting information” on page 911.

15.11 System core dump analysis

If possible, UNIX processes (including JVM process) will produce a system core dump as well as Java stack trace information in a process's working directory if it crashes. The system core dump can provide useful information as to why the process crashed, giving you a system view of a failing JVM process. However, the system core dump will not provide Java class information. Everything in the dump is C library oriented. The information provided for JVM process refers to Java's C libraries and not the reference Java class files.

15.11.1 Core dump scenario

The code samples in Example 15-21, Example 15-22, and Example 15-23 are used to show a system core dump caused by bad JNI (Java Native Interface) user applications.

Example 15-21 badjni.jsp

```
<%@ page session="false" contentType="text/html" %>
<html><body>Generating system core dump
<%
  itso.BadJni badJni = new itso.BadJni();
  badJni.badJniMethod();
%>
</body></html>
```

Example 15-22 itso.BadJni.java

```
package itso;
public class BadJni
{
    public native void badJniMethod();
    static {
        System.loadLibrary("badjni");
    }
}
```

Example 15-23 itso_BadJni.c

```
#include <jni.h>
#include "itso_BadJni.h"
#include <stdio.h>
void bad_native_method(void)
{
    char c[1];
    char *p = NULL;  /* null pointer */
    strncpy(p,c,10); /* Segmentation fault */
}
JNIEXPORT void JNICALL Java_itso_BadJni_badJniMethod
  (JNIEnv *env, jobject obj)
{
    bad_native_method();
}
```

In Example 15-23 on page 766, you can see that the program is trying to copy strings at the NULL pointer variable in the method bad_native_method. This kind of programming can crash the JVM process, producing a system core dump and Java stack trace in the process's working directory as shown in Example 15-24.

Example 15-24   System core dump and Java stack trace

```
# pwd
/usr/WebSphere/AppServer

# ls -alF *core*
-rw-r--r--   1 root system 425440811 Nov 07 19:40 core
-rw-r--r--   1 root system     60986 Nov 07 19:40 javacore22226.1036716049.txt
```

The first thing to do is to view the Java stack trace javacore file (see 15.6.1, "Understanding the Java stack trace" on page 744). The javacore file provides the Java class view of a failing JVM process as shown in Example 15-25.

Example 15-25   Java stack trace

```
Thu Nov  7 19:40:49 2002
SIGSEGV received at 0xd32a31b0 in /usr/lib/libbadjni.so. Processing terminated.
J2RE 1.3.1 IBM AIX build ca131-20020821
Current Thread Details
----------------------
    "Servlet.Engine.Transports : 9" sys_thread_t:0x44B20058
         ----- Native Stack -----
        unavailable - iar 0x4574C220 not in text area
-----------------------------------------------------------------------
...
---------------------- XM component Dump Routine  ---------------------
Full thread dump Classic VM (J2RE 1.3.1 IBM AIX build ca131-20020821, native
threads):
    "Servlet.Engine.Transports : 9" (TID:0x3210CCA0, sys_thread_t:0x44B20058,
state:R, native ID:0x2A2F) prio=5
        at itso.BadJni.badJniMethod(Native Method)
        at org.apache.jsp._badjni._jspService(_badjni.java:69)
        at com.ibm.ws.webcontainer.jsp.runtime.HttpJspBase.service(HttpJspB...)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        ...
        at com.ibm.ws.http.HttpConnection.run(HttpConnection.java(...))
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:546)
         ----- Native Stack -----
        unavailable - iar 0x4574C220 not in text area
...
```

At best, the javacore file provides a clue as to where the JVM process crashed, but only traces pure Java methods. In this example the javacore shows the itso.BadJni.badJniMethod() method of the itso.BadJni.java file to be the problem.

15.11.2 Looking at a system core dump

The core file on UNIX systems can be inspected using the dbx and gdb (GNU debugger) tools. The dbx tool is part of the AIX install (it might not be installed by default). On Sun, dbx can be installed for an additional expense. The gdb is freeware and can be downloaded.

You can issue the **dbx** command with the Java binary executable file, normally <WAS_HOME>/java/bin/java, as the parameter. The commands listed in Example 15-26 show how to find the binary executable and invoke **dbx**.

Example 15-26 Invoking dbx

```
# pwd
/usr/WebSphere/AppServer
# ls -alF core
-rw-r--r--   1 root system 425440811 Nov 07 19:40 core
# dbx /usr/WebSphere/AppServer/java/bin/java
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g
[using memory image in core]
Segmentation fault in strncpy.strncpy [/usr/lib/libbadjni.so] at 0xd32a3318
0xd32a3318 (strncpy+0x118) 9cc50001       stbu   r6,0x1(r5)
(dbx)
```

The sample was taken from an AIX 4.3.3 and AIX 5.1 system. It shows that a segmentation fault happened (SIGSEGV, signal # 11).

If you do not know where the Java binary is located, the following command will display the true Java executable name of the core:

```
# strings core | grep COMMAND_LINE
```

or

```
# strings core | more
```

After you start dbx, the **where** command provides a stack trace of where the error occurred, as shown in Example 15-27 on page 769.

Example 15-27 dbx where command

```
(dbx) where
strncpy.strncpy() at 0xd32a3318
bad_native_method() at 0xd32a315c
Java_itso_BadJni_badJniMethod(0x44b1fe4c, 0x4574c2e0) at 0xd32a31ac
mmisInvoke_V_VHelper(0x32c13ef0, 0x44b235fc, 0x1, 0x44b1fe4c, 0x4574c358) at
0xd2eb6fe4
mmipInvoke_V_V(??, ??) at 0xd2ed5e6c
```

The culprit is to be found in the bad_native_method() method of the native JNI library module. Furthermore, we can realize that the error occurred exactly when the strncpy function was executing. Look at the native JNI source code again in Example 15-23 on page 766.

And “registers” and “listi” commands are also useful to view the system registers information and the instructions.

Example 15-28 Analyzing system core dump with dbx

```
(dbx) unset $noflregs
(dbx) registers
  $r0:0x00000000  $stkp:0x4574c1c8   $toc:0x494a13c4    $r3:0x00000000
  $r4:0x4574c20b    $r5:0xffffffff    $r6:0x00000045    $r7:0x00000074
  $r8:0x000000c2    $r9:0x00000058   $r10:0x40b55a74   $r11:0x000034e0
 $r12:0x00000000   $r13:0x00000000   $r14:0x456cf800   $r15:0x4574c350
 $r16:0x44b1fe4c   $r17:0x40f0b1ec   $r18:0x00000009   $r19:0x00000000
 $r20:0x00000041   $r21:0x5604a124   $r22:0x000000c1   $r23:0x00000001
 $r24:0x32451780   $r25:0x40b20600   $r26:0x44b235fc   $r27:0x30213b08
 $r28:0x3021c118   $r29:0x44b222b4   $r30:0x44b1fe4c   $r31:0x30212418
 $iar:0xd32a3318   $msr:0x0000d0b2    $cr:0x44824844 $link:0xd32a3160
 $ctr:0x0000000a   $xer:0x00000000
         Condition status = 0:g 1:g 2:l 3:e 4:g 5:l 6:g 7:g
  $fr0:0x0000000000000000   $fr1:0x3fe8000000000000   $fr2: 0x0000000000000000
  $fr3:0x0000000000000000   $fr4:0x0000000000000000   $fr5: 0x0000000000000000
  $fr6:0x0000000000000000   $fr7:0x0000000000000000   $fr8: 0x0000000000000000
  $fr9:0x0000000000000000  $fr10:0x0000000000000000  $fr11: 0x0000000000000000
 $fr12:0x0000000000000000  $fr13:0x3fe8000000000000  $fr14: 0x0000000000000000
 $fr15:0x0000000000000000  $fr16:0x0000000000000000  $fr17: 0x0000000000000000
 $fr18:0x0000000000000000  $fr19:0x0000000000000000  $fr20: 0x0000000000000000
 $fr21:0x0000000000000000  $fr22:0x0000000000000000  $fr23: 0x0000000000000000
 $fr24:0x0000000000000000  $fr25:0x0000000000000000  $fr26: 0x0000000000000000
 $fr27:0x0000000000000000  $fr28:0x0000000000000000  $fr29: 0x0000000000000000
 $fr30:0x0000000000000000  $fr31:0x0000000000000000 $fpscr: 0xa6100000
in strncpy.strncpy [/usr/lib/libbadjni.so] at 0xd32a3318
0xd32a3318 (strncpy+0x118) 9cc50001       stbu   r6,0x1(r5)

(dbx) listi .-40,.+40
0xd32a32f4 (strncpy+0xf4) 4182009c       beq   0xd32a3390 (strncpy+0x190)
```

```
0xd32a32f8 (strncpy+0xf8) 8cc40001       lbzu   r6,0x1(r4)
0xd32a32fc (strncpy+0xfc) 8ce40001       lbzu   r7,0x1(r4)
0xd32a3300 (strncpy+0x100) 8d040001       lbzu   r8,0x1(r4)
0xd32a3304 (strncpy+0x104) 8d240001       lbzu   r9,0x1(r4)
0xd32a3308 (strncpy+0x108) 2c060000       cmpi   cr0,0x0,r6,0x0
0xd32a330c (strncpy+0x10c) 2c870000       cmpi   cr1,0x0,r7,0x0
0xd32a3310 (strncpy+0x110) 2f080000       cmpi   cr6,0x0,r8,0x0
0xd32a3314 (strncpy+0x114) 2f890000       cmpi   cr7,0x0,r9,0x0
0xd32a3318 (strncpy+0x118) 9cc50001       stbu   r6,0x1(r5)
0xd32a331c (strncpy+0x11c) 4e400020    bdzgelr
0xd32a3320 (strncpy+0x120) 4182004c        beq   0xd32a336c (strncpy+0x16c)
0xd32a3324 (strncpy+0x124) 9ce50001       stbu   r7,0x1(r5)
0xd32a3328 (strncpy+0x128) 4e400020    bdzgelr
0xd32a332c (strncpy+0x12c) 4186004c        beq  cr1,0xd32a3378 (strncpy+0x178)
0xd32a3330 (strncpy+0x130) 9d050001       stbu   r8,0x1(r5)
0xd32a3334 (strncpy+0x134) 4e400020    bdzgelr
0xd32a3338 (strncpy+0x138) 419a004c        beq  cr6,0xd32a3384 (strncpy+0x184)
0xd32a333c (strncpy+0x13c) 9d250001       stbu   r9,0x1(r5)
0xd32a3340 (strncpy+0x140) 4e400020    bdzgelr
0xd32a3344 (strncpy+0x144) 419e004c        beq  cr7,0xd32a3390 (strncpy+0x190)
(dbx) quit
```

More useful commands of dbx are:

- map
- thread
- thread info
- thread <thread_no>
- thread current <thread_no>

Type `help` for help on a command or topic and `quit` to exit dbx.

How to make sure that a good core file is generated

If, after a crash there is no core file or the output from dbx shows that the core file is truncated, ensure that:

1. The file system containing the core file has enough free space. The size required depends on your WebSphere configuration environment, but it is recommended that you have over 500 MB.
2. On AIX, ensure that “Enable full CORE dump” is set to `true` in the system environment. You can do this via smitty:

   ```
   # smitty chgsys
   ```

   Or using the following command:

   ```
   # lsattr -El sys0 | grep fullcore
   fullcore     false         Enable full CORE dump  True
   # chdev -a fullcore=true -l sys0
   ```

```
sys0 changed
# lsattr -El sys0 | grep fullcore
fullcore     true          Enable full CORE dump  True
```

3. The owner of the running process must have write permission to the directory the process dumps the core to.
4. Both of the maximum file and coredump size specifications must be large enough.

   ```
   # ulimit -a
   time(seconds)        unlimited
   file(blocks)         2097151 <-- !
   data(kbytes)         131072
   stack(kbytes)        32768
   memory(kbytes)       32768
   coredump(blocks)     2097151 <-- !
   nofiles(descriptors) 2000
   ```

   You can change the file and coredump maximum value using the following command:

   ```
   # ulimit -f nnnnnnn
   # ulimit -c nnnnnnn
   ```

   The directory where the process dumps the core is the current working directory of the process that is running. In WebSphere Application Server V5, it is normally the <WAS_HOME> directory.

15.11.3 Monitoring a running process with dbx (AIX)

Another use of **dbx** is to monitor a running process. The `-a` parameter allows the debug program to be attached to a process that is running. To attach the debug program, you need authority to use the **kill** command on this process. Use the **ps** command to determine the process ID. If you have permission, the **dbx** command interrupts the process, determines the full name of the object file, reads in the symbolic information, and prompts for commands.

Example 15-29 Monitoring a running process with dbx

```
# ps -ef|grep java
  root 18088     1 0 Nov 07  pts/2  9:06 /usr/WebSphere/... dmgr
  root 20648     1 6 Nov 07  pts/2 39:58 /usr/WebSphere/... m10df51f
  root 22234 20648 2 Nov 08  pts/2 32:19 /usr/WebSphere/... server1

# dbx -a 22234
Waiting to attach to process 22234 ...
Successfully attached to java.
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g
```

```
stopped in _event_sleep at 0xd00549dc
0xd00549dc (_event_sleep+0x90) 80410014       lwz   r2,0x14(r1)
(dbx) where
_event_sleep(??, ??, ??, ??, ??) at 0xd00549dc
_event_wait(??) at 0xd0054ec8
_cond_wait_local(??, ??, ??) at 0xd0060e04
_cond_wait(??, ??, ??) at 0xd0061298
pthread_cond_wait(??, ??) at 0xd0061fbc
condvarWait(??, ??, ??) at 0xd3056160
sysMonitorWait(??, ??, ??, ??) at 0xd305511c
lkMonitorWait(??, ??, ??, ??) at 0xd2f39724
JVM_MonitorWait(??, ??, ??, ??) at 0xd2eab14c
(dbx) detach
#
```

To continue execution of the application and exit dbx, type `detach` instead of `quit`. (If you typed quit to exit, the process would stop running.)

Note: Do not use the command **`dbx -a <pid>`** on heavily loaded systems because it could be temporarily blocked.

Note: In 15.6.1, "Understanding the Java stack trace" on page 744, we explained how to dump Java stack traces using Thread Analyzer or the **`kill`** command. If a process appears to hang, it is probably a good idea to view both a Java stack trace and a system thread dump:

1. To generate the Java stack trace (generates a javacore file):

   ```
   #kill -3 <java_pid>
   ```

2. To view the system thread dump:

   ```
   #dbx -a <pid>
   ```

15.11.4 errpt command (AIX)

On AIX, the **`errpt`** command generates an error report from entries in a system error log. If you have a system core dump, the event would be logged in the system error log. You can check it as follows.

Example 15-30 errpt command

```
# errpt -a > /tmp/errpt.txt
# vi /tmp/errpt.txt
....
---------------------------------------------------------------------------
LABEL:          CORE_DUMP
```

```
IDENTIFIER:     C60BB505

Date/Time:       Thu Nov  7 19:40:49 EST
...
Detail Data
SIGNAL NUMBER
          11
USER'S PROCESS ID:
       22226
...
PROGRAM NAME
java
ADDITIONAL INFORMATION
strncpy 118
bad_nativ 20
Java_itso 18
mmisInvok 2A4
entryCmp FFFFE688
??
...
SYMPTOM CODE
PCSS/SPI2 FLDS/java SIG/11 FLDS/strncpy VALU/118 FLDS/bad_nativ
---------------------------------------------------------------------------
...
```

A portion of the name of the native methods processing at the time of the JVM crash is shown in the error report. Even though the **errpt** command does not provide the fully qualified name, it is enough to start tracing the problem.

15.11.5 Core files on the Windows platform

Windows Dr. Watson log files are similar to core files on UNIX. To find out about the format of Windows Dr. Watson log files:

1. Open a command prompt and enter the following command:

   ```
   C:\>drwtsn32
   ```

2. Click **Help -> Dr. Watson log file overview**.

15.12 Application debugging and tracing

Debugging applications is beyond the scope of this book. However, we want to point out two facilities provided by WebSphere Application Server:

- Application Server Toolkit
- JRas extensions and logging toolkit

15.12.1 Application Server Toolkit

The Application Server Toolkit is included with WebSphere Application Server V5. It includes debugging functionality built on the Eclipse workbench. It includes the following:

- The WebSphere Application Server debug adapter:

  Allows you to debug Web objects that are running on WebSphere Application Server and that you have launched in a browser. These objects include EJBs, JSPs, and servlets.

- The JavaScript debug adapter:

  Enables server-side JavaScript debugging.

- The compiled language debugger:

  Allows you to detect and diagnose errors in compiled-language applications.

- The Java development tools (JDT) debugger:

  Allows you to debug Java.

All debug components in the Application Server Toolkit can be used for both local and remote debugging. To learn more about the debug components, launch the Application Server Toolkit and select **Help -> Help Contents**. Choose the **Debugger Guide** bookshelf entry.

15.12.2 JRas logging toolkit and JRas extensions

WebSphere Application Server provides a message logging and diagnostic trace API that can be used by applications. This API is based on the stand-alone JRas logging toolkit developed by IBM. The stand-alone JRas logging toolkit is a collection of interfaces and classes that provide message logging and diagnostic trace primitives. These primitives are not tied to any particular product or platform. The stand-alone JRas logging toolkit provides a limited amount of support (typically referred to as systems management support), including log file configuration support based on property files.

As designed, the stand-alone JRas logging toolkit does not contain the support required for integration into the WebSphere Application Server runtime or for usage in a J2EE environment. To overcome these limitations, WebSphere Application Server provides a set of extension classes to address these shortcomings. This collection of extension classes is referred to as the JRas extensions. The JRas extensions do not modify the interfaces introduced by the stand-alone JRas logging toolkit, but simply provide the appropriate implementation classes.

15.13 Product installation information

The WebSphere Application Server version and the versions of related software are important. All components need to be at the correct versions for proper interoperation. In this section we describe how to determine the versions and build levels of the various components in your environment.

15.13.1 Using the administrative console to find product information

Note: This method can only be used if the application server is running.

Perhaps the easiest way to get comprehensive information about the installation is to use the administrative console. You can use this when the server is running.

1. Select **Servers -> Application Servers**.
2. Click the server.
3. Select the **Runtime** tab.
4. Click **Product Information**.

15.13.2 Locating WebSphere Application Server version information

To check the version of the product installed on each of the nodes in the cell and the Network Deployment product installed in the dmgr node, use one of these option:

1. Look in the product version properties file of the specific installation:

Base:	<WAS_HOME>/properties/version/BASE.product
Deployment manager:	<WAS_ND_HOME>/properties/version/ND.product

   The file will contain content similar to that shown in Example 15-31.

Example 15-31 BASE.product content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE product PUBLIC "productId" "product.dtd">
<product name="IBM WebSphere Application Server">
  <id>BASE</id>
  <version>5.1.0</version>
  <build-info date="11/4/03" level="b0344.02"></build-info>
</product>
```

2. Look in the SystemOut.log file of the specific installation:

   Base: <WAS_HOME>/logs/nodeagent/SystemOut.log

   Deployment manager: <WAS_ND_HOME>/logs/dmgr/SystemOut.log

   The file will contain content similar to that shown in Example 15-32.

Example 15-32 Node agent SystemOut.log content

```
************ Start Display Current Environment ************
WebSphere Platform 5.1 [BASE 5.1.0 b0344.02] [ND 5.1.0 b0344.02]  running with
process name Net1Network\Net1_JH\Net1_JH and process id 1912
Host Operating System is Windows 2000, version 5.1
Java version = J2RE 1.4.1 IBM Windows 32 build cn1411-20031011 (JIT enabled:
jitc), Java Compiler = jitc, Java VM name = Classic VM
...
...
************* End Display Current Environment *************
```

3. Execute the **versionInfo** command from the bin directory of the installation. Output similar to Example 15-33 will be generated.

Example 15-33 versionInfo output for base installation

```
$ cd <WAS_HOME>/bin
$ versionInfo
------------------------------------------------------------------------
IBM WebSphere Application Server Version Report
------------------------------------------------------------------------
   Platform Information
------------------------------------------------------------------------
         Name: IBM WebSphere Application Server
         Version: 5.1

   Product Information
------------------------------------------------------------------------
         ID: BASE
         Name: IBM WebSphere Application Server
         Build Date: 11/4/03
         Build Level: b0344.02
         Version: 5.1.0
   Product Information
------------------------------------------------------------------------
         ID: ND
         Name: IBM WebSphere Application Server for Network Deployment
         Build Date:11/4/03
         Build Level: b0344.02
         Version: 5.1.0
------------------------------------------------------------------------
End Report
------------------------------------------------------------------------
```

Note: When executed from the Deployment Manager bin directory, **versionInfo** only lists the Deployment Manager version details. When executed from the bin directory of any of the nodes, both the local node's and the Deployment Manager version details are listed.

15.13.3 Finding the JDK version

To determine which version of the JDK is installed in your environment, use one of the following options:

1. Look in the SystemOut.log file:

Base:	<WAS_HOME>/logs/nodeagent/SystemOut
Deployment manager:	<WAS_ND_HOME>/logs/dmgr/SystemOut

2. Run **java -fullversion** from the command line:

```
<WAS_HOME>/java/bin/java -fullversion
```

15.13.4 Finding the IBM HTTP Server version

To check the version of your IBM HTTP Server on Windows platforms, run **apache -v**. For example:

```
"D:\IBM HTTP Server\apache.exe -v”
Server version: IBM_HTTP_SERVER/1.3.28 Apache/1.3.28 (Win32)
Server built: Oct 9 2003 16:16:55
```

On UNIX platforms, run **httpd -v**.

15.14 Resources for problem determination

- WebSphere Application Server support (FixPak, fixes, and hints and tips)

  `http://www-3.ibm.com/software/webservers/appserv/support.html`
- IBM alphaWorks® emerging technologies

  `http://www.alphaworks.ibm.com`
- IBM developerWorks™

  `http://www.ibm.com/developerworks/`
- Worldwide WebSphere User Group

  `http://www.websphere.org`
- Google search group: WebSphere Application Server

  `http://groups.google.com/groups?group=ibm.software.websphere.application-server`
- *An Introduction to JavaTM Stack Traces*

  `http://developer.java.sun.com/developer/technicalArticles/Programming/Stacktrace/`
- *Apache HTTP Server Log Files* at:

  `http://httpd.apache.org/docs/logs.html`

16

Command-line administration and scripting

In this chapter, we introduce the WebSphere scripting solution called wsadmin and describe how some of the basic tasks that are performed by WebSphere Administrators can be done using the scripting solution. There are two types of tasks: the operational task and the configurational task. We cover both types of tasks using wsadmin. The operational tasks deal with currently running objects in WebSphere installation and the configurational tasks deal with the configuration of WebSphere installations.

This chapter contains the following:

- Overview of scripting concept
- Overview of wsadmin basics
- Common operational administration tasks using wsadmin
- Common configurational administration tasks using wsadmin
- Case study of managing the Webbank application using wsadmin
- Migration of WebSphere V4.0 WSCP scripts

In WebSphere Application Server V5.1, both the Jacl and Jython scripting languages are supported by the wsadmin tool. This chapter shows scripting using Jacl. Jython scripting looks similar; there are many examples in the InfoCenter that you can use to get started with Jython.

16.1 Overview of WebSphere scripting

WebSphere Application Server V5 provides a new scripting interface based on the Bean Scripting Framework (BSF) called wsadmin. BSF is an open source project to implement an architecture for incorporating scripting into Java applications and applets. The BSF architecture works as an interface between Java applications and scripting languages. Using BSF allows scripting languages to do the following:

- Look up a pre-registered bean and access a pre-declared bean
- Register a newly created bean
- Do all bean operations
- Bind events to scripts in the scripting language

Figure 16-1 shows the major components involved in the wsadmin scripting solution.

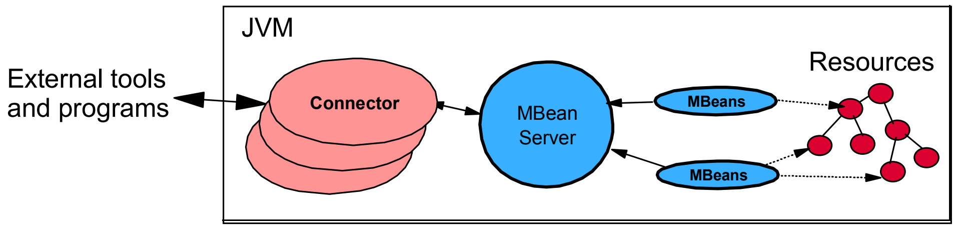


Figure 16-1 wsadmin scripting

Since wsadmin uses BSF, it can make various Java objects available through language-specific interfaces to scripts. There are four objects available to scripts:

- AdminControl: Used to invoke operational commands.
- AdminConfig: Used to invoke configurational command to create or modify WebSphere configurational elements.
- AdminApp: Used for administering applications.
- Help: Used for general help.

The scripts use those objects to communicate with MBeans running in WebSphere server processes. Supported scripting languages all have ways to invoke methods on the exposed Java objects.

16.2 Java Management Extensions (JMX)

Java Management Extensions (JMX) is a framework that provides a standard way of exposing Java resources (application servers, for example) to a system management infrastructure. The JMX framework allows a provider to implement functions, such as listing the configuration settings, and allows users to edit the settings. It also includes a notification layer that can be used by management applications to monitor events such as the startup of an application server.

The use of JMX opens the door to third-party management tool providers. Users of WebSphere are no longer restricted to IBM-supplied management tools.

JMX is a Java specification (JSR-003) that is part of J2SE 1.4.

Note: Understanding JMX is not really necessary in order to use the WebSphere scripting tool, wsadmin. This section provides some good background information about JMX, but if you are simply interested in using wsadmin, skip to 16.3, “Using wsadmin” on page 788.

16.2.1 JMX key features

The key features of the WebSphere Application Server V5 implementation of JMX include:

- All processes run the JMX agent.
- All runtime administration is performed through JMX operations.
- Connectors are used to connect a JMX agent to a remote JMX-enabled management application. The following connectors are currently supported:
  - SOAP JMX Connector
  - RMI/IIOP JMX Connector
- Protocol adapters provide a management view of the JMX agent through a given protocol. Management applications that connect to a protocol adapter are usually specific to a given protocol.
- Allows a runtime object’s configuration settings to be queried and updated.
- Allows application components and resources to be loaded, initialized, changed and monitored in the runtime.

16.2.2 JMX benefits

The use of JMX for management functions in WebSphere Application Server provides the following benefits:

- Enables Java applications to be managed without heavy investment.

  Relies on a core managed object server that acts as a management agent. Java applications simply need to embed a managed object server and make some of its functionality available as one or several MBeans registered with the object server.

- Provides a scalable management architecture.
  - Every JMX agent service is an independent module that can be plugged into the management agent.
  - The API is extensible, allowing new WebSphere and custom application features to be easily added and exposed through this management interface.
  - Integrates existing management solutions.
  - JMX smart agents are capable of being managed through HTML browsers or by various management protocols such as Web services, JMS, and SNMP.
  - Each process is self-sufficient when it comes to the management of its resources. There is no central point of control. In principle, a JMX-enabled management client could be connected to any managed process and interact with the MBeans hosted by that process.
  - JMX does allow a single, flat, domain-wide approach to system management. Separate processes interact through MBean proxies allowing a single management client to seamlessly navigate through a network of managed processes.
- Defines only the interfaces necessary for management.
- Provides a standard API for exposing application and administrative resources to management tools.

16.2.3 JMX architecture

The JMX architecture is structured into three layers:

- Instrumentation layer

  Dictates how resources can be wrapped within special Java beans, called Management Beans (MBeans).
- Agent layer

  Consists of the MBean server and agents, which provide a management infrastructure. Services implemented include:
  - Monitoring
  - Event notification
  - Timers

- Management layer

  Defines how external management applications can interact with the underlying layers in terms of protocols, APIs, etc. This layer uses an implementation of the *Distributed Services* specification (JSR-077), which is not yet part of the J2EE specification

The layered architecture of JMX is summarized in Figure 16-2.

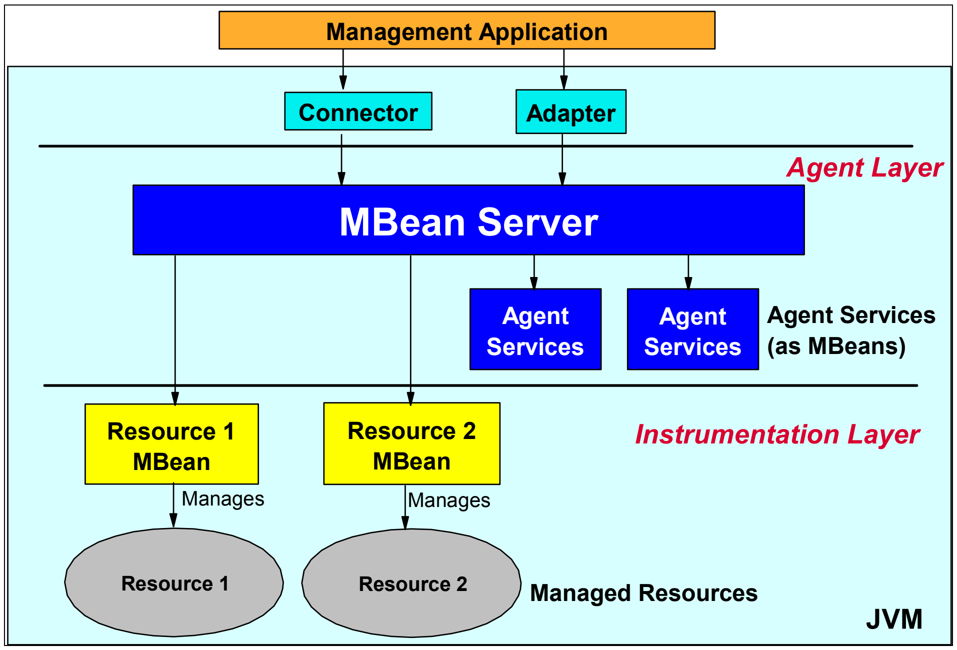


Figure 16-2 JMX architecture

How does JMX work?

Resources are managed by JMX MBeans. These are not EJBs, but simple JavaBeans that need to conform to certain design patterns outlined in the JMX specification.

Providers that want to instrument their systems with JMX need to provide a series of MBeans. Each MBean is meant to wrap (or represent) a certain runtime resource. For instance, in order to expose an application server as a manageable resource, WebSphere needs to provide an application server MBean.

External applications can interact with the MBeans through the use of JMX connectors and protocol adapters, as shown in Figure 16-1 on page 780.

Connectors are used to connect an agent with a remote JMX-enabled management application. This form of communication involves a connector in the JMX agent and a connector client in the management application.

The key features of JMX connectors are:

- Connectors are oriented to the transport mechanism. For example, a provider may provide an RMI connector that allows Java applications to interact remotely with the MBeans.
- The connector translates JavaBeans calls to the a protocol stream.
- There is a 1:1 mapping between client method invocations and MBean operations.
- This is the low-level API for accessing MBeans.

Notes:

1. Connectors cannot be used with existing non-JMX aware management applications, as connectors communicate “native JMX” information.
2. The RMI connector requires that the ORB be configured and running in the process. The JMS server does not need the ORB (other than if the RMI connector were desired) and the ORB adds a 30 MB overhead to the process. Therefore the default configuration for the JMS server is not to include the ORB and therefore the RMI connector cannot be configured either.

Protocol adapters

Protocol adapters provide a management view of the JMX agent through a given protocol. Management applications that connect to a protocol adapter are usually specific to the given protocol.

The key features of JMX protocol adapters are:

- Protocol adapters adapt operations of MBeans and the MBean server into a representation in the given protocol, and possibly into a different information model, for example SNMP or HTTP.
- There is not a 1:1 mapping between client method invocations and MBean operations.
- This is the high-level API for accessing MBeans.

MBean server

Each JMX enabled JVM contains an MBean server that registers all the MBeans in the system. It is the MBean server that provides access to all of its registered MBeans.

Note: There is only one MBean server per JVM.

Both connectors and protocol adapters use the services of the MBean server in order to apply the management operation they receive to the MBeans, and in order to forward notifications to the management system. Connector and protocol adapter communication is summarized in Figure 16-3.

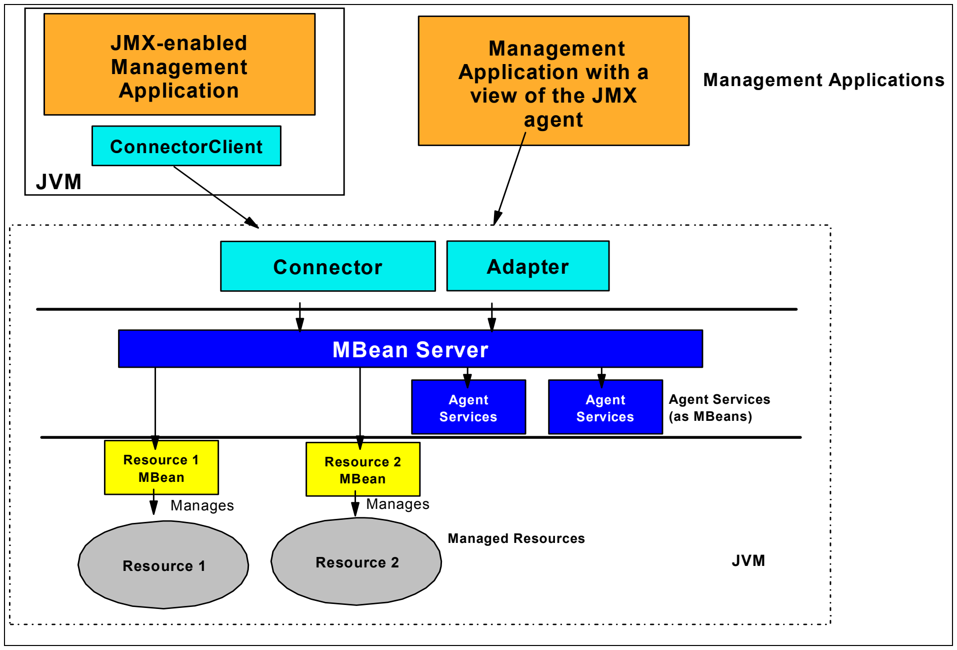


Figure 16-3 JMX connectors and adapters

16.2.4 JMX distributed administration

Figure 16-4 on page 786 shows how the JMX architecture fits into the overall distributed administration topology of a Network Deployment environment.

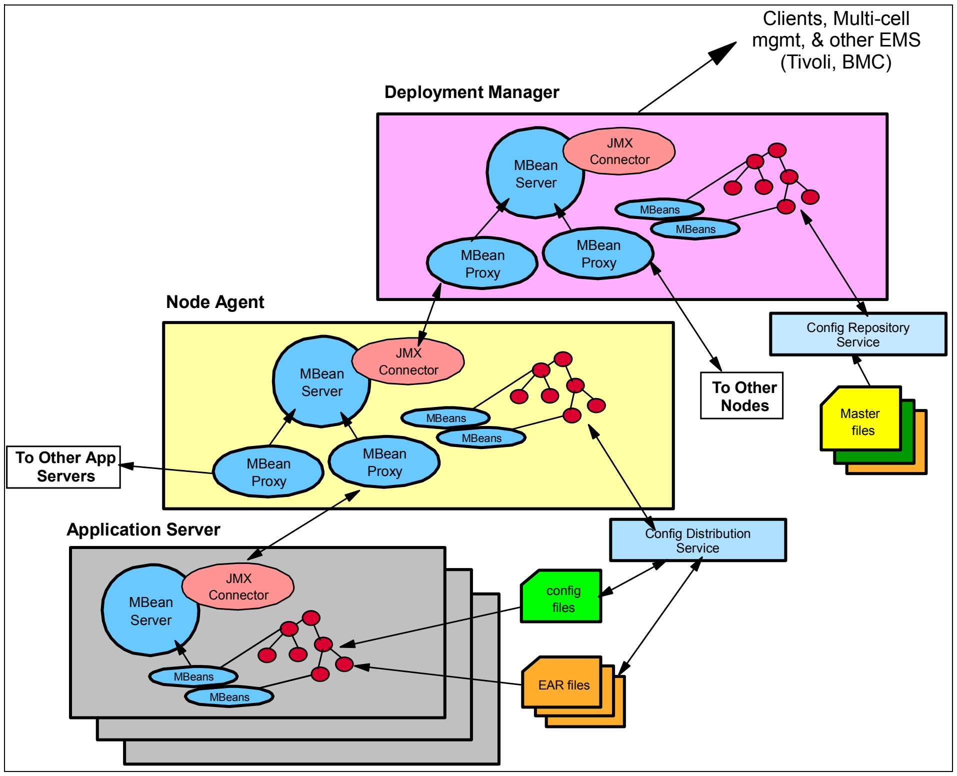


Figure 16-4 JMX distributed administration

The key points of this distributed administration architecture are:

- Internal MBeans (local to the JVM) register with the local MBean server.
- External MBeans have a local proxy to their MBean server. The proxy registers with the local MBean server. The MBean proxy allows the local MBean server to pass the message to an external MBean server located on:
  - Another server
  - Node agent
  - Deployment Manager
- A node agent has an MBean proxy for all servers within its node. However, MBean proxies for other nodes are not used.
- The Deployment Manager has MBean proxies for all node agents in the cell.

The configuration of MBean proxies is shown in Figure 16-5 on page 787.

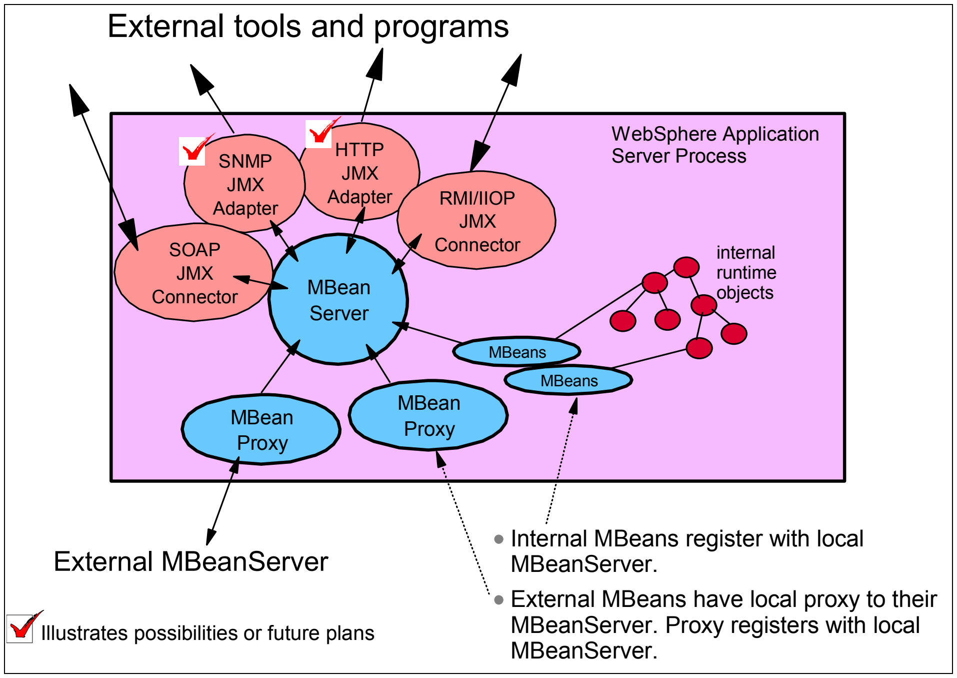

External tools and programs
WebSphere Application Server Process
SNMP JMX Adapter
HTTP JMX Adapter
RMI/IIOP JMX Connector
SOAP JMX Connector
MBean Server
internal runtime objects
MBeans
MBeans
MBean Proxy
MBean Proxy
External MBeanServer
Internal MBeans register with local MBeanServer.
External MBeans have local proxy to their MBeanServer. Proxy registers with local MBeanServer.
Illustrates possibilities or future plans

Figure 16-5 JMX architecture

16.2.5 JMX MBeans

WebSphere Application Server provides a number of MBeans, each of which may have different functions/operations available. For example:

- An application server MBean may expose operations such as start and stop.
- An application MBean may expose operations such as install and uninstall.

16.2.6 JMX usage scenarios

Some of the more common JMX usage scenarios you will encounter are:

- Internal product usage:

  All WebSphere Application Server V5 administration clients use JMX:

  – WebSphere administrative console.
  – wsadmin scripting client
  – Admin client Java API.

- External programmatic administration

  In general, most external users will not be exposed to the use of JMX. Instead, they will access administration functions through the standard WebSphere Application Server V5 administration clients.

  However, external users would need to access JMX in the following scenarios:

  – External programs written to control the Network Deployment runtime and its WebSphere resources by programmatically accessing the JMX API.
  – Third-party applications that include custom JMX MBeans as part of their deployed code, allowing the applications components and resources to be managed via the JMX API.

16.3 Using wsadmin

In this section, we describe what needs to be configured for launching wsadmin, how to start wsadmin, and getting online information.

16.3.1 Configuring wsadmin

The properties that determine the scripting environment for wsadmin can be set using either the command line or a properties file. Properties can be set in the following three ways:

- Using the system default properties file:

  `<WAS_HOME>/properties/wsadmin.properties`
- Using a customized properties file placed in the user home directory or in $user_home. You can copy the default properties file to this location and modify it.
- By starting wsadmin on the command line.

The properties to note are:

com.ibm.ws.scripting.connectionType:	SOAP, RMI or JMX
com.ibm.scripting.host:	host name of system
com.ibm.ws.scripting.defaultLang:	jacl, JavaScript or Jython
com.ibm.ws.scripting.traceFile:	File for trace information
com.ibm.ws.scripting.traceString:	=com.ibm.*=all=enabled

Some of the listed properties in the wsadmin.properties file are commented out by default. An example is com.ibm.ws.scripting.traceString. If you want to trace wsadmin execution, remove the comment sign # from the properties file.

Similarly some of the properties contain values. For example, com.ibm.ws.scripting.connectionType has a default value of SOAP. This means that when a scripting process is invoked, a SOAP connector is used to communicate with the server.

In addition to the properties file, you should also take note of the profile file. A profile is a script that is invoked before the main script or before invoking wsadmin in interactive mode. The purpose of the profile is to customize the environment in which scripts run. For example, a profile can be set for Jacl scripting language that makes Jacl-specific variables or procedures available to the interactive session or main script.

16.3.2 Launching wsadmin

The wsadmin .bat (Windows) or .sh (UNIX) resides in the bin directory of the WebSphere Application Server or Network Deployment installation and can be started from a command prompt with the command:

```
<WAS_HOME>\bin\wsadmin.bat (.sh)
```

To get syntax-related help, type `wsadmin.bat -?` and press **Enter**. Example 16-1 shows the output. Some of these options have an equivalent in the properties file. Any options specified on the command line will override those set in the properties file.

Example 16-1 wsadmin command-line options

```
C:\WebSphere\DeploymentManager\bin>wsadmin -h
WASX7001I: wsadmin is the the executable for WebSphere scripting.
Syntax:

wsadmin
        [ -h(elp)  ]
        [ -?  ]
        [ -c <command> ]
        [ -p <properties_file_name>]
        [ -profile <profile_script_name>]
        [ -f <script_file_name>]
        [ -javaoption java_option]
        [ -lang  language]
        [ -wsadmin_classpath  classpath]
        [ -conntype
                SOAP
                        [-host host_name]
                        [-port port_number]
                        [-user userid]
                        [-password password] |
                RMI
```

```
                        [-host host_name]
                        [-port port_number]
                        [-user userid]
                        [-password password] |
                    JMS <jms parms> |
                    NONE
                ]
                [ script parameters ]
```

Running a single command (-c)

The `-c` option is used to execute a single command using wsadmin.
Example 16-2 shows the usage of single command execution using wsadmin. In the example we use the AdminControl object to query the node name of the WebSphere server process.

Example 16-2 Running a single command in wsadmin

```
C:\PROGRA~1\WEBSPH~1\APPSER~1\bin>wsadmin -c "$AdminControl getNode"

WASX7209I: Connected to process "server1" on node MyServ using SOAP connector;
The type of process is: UnManagedProcess
MyServ
```

Running script files (-f)

The `-f` option is used to execute a script file. Example 16-3 shows a two-line Jacl script named myScript.jacl. The script has a .jacl extension (.jacl), letting wsadmin know it is an Jacl script. If there is no extension, the com.ibm.ws.scripting.defaultLang property is used to determine the language. If this setting is not correct, use the `-lang` option to identify the scripting language.

Example 16-3 Jacl script

```
puts "This is an example JACL script"
puts "[$AdminControl getNode]"
```

Example 16-4 shows how to execute the script.

Example 16-4 Running a script in wsadmin

```
C:\Program Files\WebSphere\AppServer\bin>wsadmin -f c:\test\myScript.jacl

WASX7209I: Connected to process "server1" on node MyServ using SOAP connector;
The type of process is: UnManagedProcess

This is an example JACL script
MyServ
```

Using a profile (-profile)

The `-profile` command line option can be used to specify a profile script. The profile can be used to perform whatever standard initialization is required. Several `-profile` options can be used on the command line and those are invoked in the order given.

Specifying a properties file (-p)

The `-p` option is used to specify a properties file other than wsadmin.properties either located in the <WAS_HOME>/properties directory or in the $user_home directory.

Figure 16-5 shows an example of invoking wsadmin to execute a script file using a specific properties file.

Example 16-5 Specifying properties file on the command line

```
C:\Program Files\WebSphere\AppServer\bin>wsadmin -f
c:\scriptexamples\myScript.jacl -p c:\temp\custom.properties

WASX7209I: Connected to process "server1" on node MyServ using SOAP connector;
The type of process is: UnMana
gedProcess

This is an example JACL script
MyServ
```

Getting help for wsadmin objects

wsadmin exposes four different objects used to manage WebSphere from the command line:

- AdminControl
- AdminConfig
- AdminApp
- Help

When writing wsadmin scripts, the Help object provides information about the available methods for these objects. For example, to get a list of the public methods available for the AdminControl object, enter the following command:

```
   wsadmin -c "$Help AdminControl"
```

Similarly, you can do this for the AdminConfig, AdminApp, and Help objects.

Note: For the purposes of this discussion, we will refer to the methods of the AdminControl, AdminConfig, AdminApp, and Help objects as “commands”.

Getting help for a specific command

You can find help for any command provided by the wsadmin objects using the following syntax:

```
wsadmin> <wsadmin_object> help <method_name>
```

Example 16-6 shows how to obtain help information for the completeObjectName command of the AdminControl object.

Example 16-6 Getting help for a command

```
wsadmin>$AdminControl help completeObjectName
WASX7049I: Method: completeObjectName

        Arguments: object name, template

        Description: Returns a String version of an object name that matches
        the "template."  For example, the template might be "type=Server,*"
        If there are several MBeans that match the template, the first match
        is returned.
```

Finding information for running MBeans

MBeans represent running objects in WebSphere. Each server contains an MBean server.

You can use queryNames to discover what MBeans are running. The simplest form of this command in Jacl is as follows:

```
$AdminControl queryNames *
```

This returns a list of all MBeans of all types to the MBean server. Depending on the server your scripting client is attached to, this list may contain MBeans that are running in many different servers.

- If the client is attached to a stand-alone WebSphere Application Server, the list will contain only MBeans running on that server.
- If the client is attached to a node agent, the list will contain MBeans running in the node agent as well as MBeans running on all application servers on that node.
- If the client is attached to a Deployment Manager, the list will contain MBeans running in the Deployment Manager, in all node agents communicating with that Deployment Manager, and all application servers on all the nodes served by those node agents.

Example 16-7 on page 793 shows a Jacl script that collects information about running MBeans into a file called mbean.txt. In the output file, the MBean information is preceded by a label called “ObjectName” to clearly identify the information.

The script is written for execution on Windows. Before running the script, a file named mbean.txt needs to be created in the temp folder. To run this script on a UNIX platform, change the path to mbean.txt file to /tmp/mbean.txt.

Example 16-7 Finding information for running MBeans

```
set file "c:\\temp\\mbean.txt"
set logFile [open $file a]
set mblist [$AdminControl queryNames "*:*"]
foreach item $mblist {
puts $logFile "ObjectName: $item"
}
close $logFile
```

The items comprising the returned list are string representations of JMX ObjectName objects. For example:

```
WebSphere:cell=MyServNetwork,name=dmgr,mbeanIdentifier=server.xml#Server_1,type
=Server,node=MyServManager,process=dmgr,processType=DeploymentManager
```

This represents a Deployment Manager (dmgr) running in cell MyServ on node MyServManager. WebSphere includes the following key properties on its object names:

- Name
- Type
- Cell
- Node
- Process
- mbeanIdentifier

You can use any of these key properties to narrow the scope of the queryNames. For example you can list all MBeans that represent “Server” objects on the node myNode, as follows:

```
$AdminControl queryNames WebSphere:type=Server,node=myNode,*
```

Notes:

- You will get an empty list back if you do not use the `*` wildcard at the end of the ObjectName.
- `WebSphere:` represents the domain and is assumed if you do not include it.

Finding attributes and operations for running MBeans

The Help object can be used to find information about any running MBeans in the system. The easiest way is to use either the **attributes** or **operations** command. You first must get the ObjectName for the running MBeans. This can be done using the AdminControl completeObjectName command. Then you can use the Help object to examine the attributes.

Example 16-8 shows how to find attributes information for a running server MBean. The first command initializes a variable called "serv" to the completeObjectName of the servers running on the node. The **attributes** command of the Help object lists all the available attributes for the MBean.

Example 16-8 Finding attributes for a running MBean

```
wsadmin>set serv [$AdminControl completeObjectName type=Server,node=MyServ,*]

WebSphere:cell=MyServNetwork,name=MyServ,mbeanIdentifier=server.xml#Server_1,ty
pe=Server,node=MyServ,process=MyServ,processType=NodeAgent

wsadmin>$Help attributes $serv

Attribute                      Type                           Access
name                           java.lang.String               RO
pid                            java.lang.String               RO
cellName                       java.lang.String               RO
deployedObjects                [Ljava.lang.String;            RO
javaVMs                        [Ljava.lang.String;            RO
nodeName                       java.lang.String               RO
processType                    java.lang.String               RO
resources                      [Ljava.lang.String;            RO
serverVersion                  java.lang.String               RO
serverVendor                   java.lang.String               RO
state                          java.lang.String               RO
platformName                   java.lang.String               RO
platformVersion                java.lang.String               RO
```

Similarly you can use the **operations** command to find out what operations are supported by this MBean. Example 16-9 shows the usage of operation command and its output.

Example 16-9 Finding operations information for a running MBean

```
wsadmin>$Help operations $serv
Operation
java.lang.String getName()
java.lang.String getPid()
java.lang.String getCellName()
[Ljava.lang.String; getDeployedObjects()
```

```
[Ljava.lang.String; getJavaVMs()
java.lang.String getNodeName()
java.lang.String getProcessType()
[Ljava.lang.String; getResources()
java.lang.String getServerVersion()
java.lang.String getServerVendor()
java.lang.String getState()
java.lang.String getPlatformName()
java.lang.String getPlatformVersion()
java.lang.String getProductVersion(java.lang.String)
java.lang.String getComponentVersion(java.lang.String)
java.lang.String getEFixVersion(java.lang.String)
java.lang.String getExtensionVersion(java.lang.String)
```

Finding information about configuration objects

Prior to making changes to a configuration, you need to know what objects are available, their default value, and the parent for a particular object. The AdminConfig object provides commands that can be used to get the basic information.

types

The **`types`** command returns a list of configuration object types a user can manipulate. Example 16-10 shows the partial output of the **`types`** command. Each object type in the list can be created or modified.

Example 16-10 Output of types command

```
wsadmin>$AdminConfig types

AdminService
Agent
ApplicationConfig
ApplicationContainer
ApplicationDeployment
ApplicationServer
AuthMechanism
AuthenticationTarget
AuthorizationConfig
AuthorizationProvider
AuthorizationTableImpl
BackupCluster
CMPConnectorFactory
CORBAObjectNameSpaceBinding
Cell
CellManager
Classloader
ClusterMember
```

```
ClusteredTarget
CommonSecureInterop
Component
ConfigSynchronizationService
```

getid

The **getid** command returns the configuration ID for an object. Configuration objects are named using a convention that uses a combination of the display name for the object and its configuration ID. The ID uniquely identifies an object and can be used in any configuration command that requires an object name.

Example 16-11 shows how to obtain the configuration ID for server1. The string argument passed to the command identifies the node and server to get the ID for. The "**/**" is used to separate one set of object type/value from another. The "**:**" is used to separate the value from the object type in an object type and value pair.

Example 16-11 Finding configuration information of an object

```
wsadmin>$AdminConfig getid "/Node:MyServ/Server:server1/"

server1(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#Server_1)
```

Note: Configuration objects are named using a combination of the "display name" and its configuration ID. The display name comes first, followed by the configuration ID in parentheses. An example of such an object name is:

```
server1(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#Server_1
)
```

For those pieces of configuration data that do not have display names, the name of the object simply consists of the configuration ID in parentheses. An example of such an object name is as follows:

```
(cells/testcell/nodes/testnodes/resource.xml#J2CSecurityPermission_1)
```

Since the ID portion is completely unique, a user can always use it without the prepended display name in any command that requires a config object name.

List

The **list** command returns a list of objects of a given type. In a WebSphere Application Server environment, there are several object types and there are many objects configured that have the same object type. You can display all the objects of the same object type using the **list** command.

Example 16-12 list all objects of DynamicCache object type in our test environment. You can see the **list** command returns two objects of DynamicCache type, one for server1 and another for the WebbankAS server.

Example 16-12 Finding objects of the same object type

```
wsadmin>$AdminConfig list DynamicCache

(cells/MyServNetwork/nodes/MyServ/servers/WebbankAS:server.xml#DynamicCache_1)
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#DynamicCache_1)
```

Defaults

The **defaults** command returns a table of attributes, their types and defaults if any. Each object has an object type and each object type has attributes that may or may not have default values.

Example 16-13 shows the usage of the **default** command to list the attributes and default values for those attributes for object type DynamicCache.

Example 16-13 Finding attributes and default values for an object type

```
wsadmin>$AdminConfig defaults DynamicCache
Attribute                        Type                             Default
enable                           Boolean
cacheSize                        Integer                          2000
defaultPriority                  Integer                          1
replicationType                  ENUM
pushFrequency                    Integer                          0
enableDiskOffload                Boolean                          false
diskOffloadLocation              String
hashSize                         Integer
context                          ServiceContext
properties                       Property
cacheGroups                      ExternalCacheGroup
cacheReplication                 DRSSettings
```

parents

The **parents** command returns a list of object types that can serve as the parent of the given type. An object can contain other objects. Therefore a parent child relationship exists in the configuration. For example a node type object contains server type objects, making the node object a parent to the server objects.

In Example 16-11 on page 796 we used the **getid** command to obtain the configuration ID for server1. As input to the command, we passed the value of the Node type object type along with the value of the Server type object. In order to know which object type can be used as a parent type object, you can use the

parents command as shown in Example 16-14. The output of the command shows us that Node is the parent object for Server.

Example 16-14 Finding the parent type objects for an object type

```
wsadmin>$AdminConfig parents Server
Node
```

Input and output of configuration object attributes

The AdminConfig **attributes** command is part of the wsadmin online help feature. The information displayed does not represent any particular configuration object but represents configuration object types or object “meta-data”. The meta-data is used to show, modify, and create actual configuration objects. In this section we describe how to interpret the output of those commands.

The **attributes** command displays the type and name of each attribute defined for a given type of configuration object. The list of attributes is displayed slightly differently in different languages. In Jacl, each attribute representation is separated by commas, so the output can be treated as a Tcl list. The name of each attribute is always a string, generally beginning with a lowercase letter. But the types of attributes vary. We use an example to show various types of attributes.

Example 16-15 shows the output of the **attributes** command for configurational object called DynamicCache. There are 12 attributes listed. You can see there are four simple integer attributes, two Boolean attributes and one String attribute.

The cacheGroups and properties objects are lists of objects indicated by * at the end of ExternalCacheGroup and Property(TypedProperty) respectively. These are nested attributes. Another **attribute** command can be used to see the composition of these nested attributes.

Example 16-15 Output of attribute command of AdminConfig object

```
wsadmin>$AdminConfig attributes DynamicCache

"cacheGroups ExternalCacheGroup*"
"cacheReplication DRSSettings"
"cacheSize Integer"
"context ServiceContext@"
"defaultPriority Integer"
"diskOffloadLocation String"
"enable Boolean"
"enableDiskOffload Boolean"
"hashSize Integer"
"properties Property(TypedProperty)*"
```

```
"pushFrequency Integer"
"replicationType ENUM(PULL, PUSH, PUSH_PULL, NONE)"

wsadmin>$AdminConfig attributes ExternalCacheGroup
"members ExternalCacheGroupMember*"
"name String"
"type ENUM(SHARED, NOT_SHARED)"

wsadmin>$AdminConfig attributes TypedProperty

"description String"
"name String"
"required Boolean"
"type String"
"validationExpression String"
"value String"
```

In Example 16-15 on page 798, you can see that the properties attribute has a value that is also a list of objects of the Property type. The Property type is a generic type, so its sub-types are listed, that is TypedProperty. The replicationType attribute is an ENUM type attribute whose value must be one of the four strings provided in parentheses.

The **show** command of the AdminConfig object can be used to display the top-level attributes of a given object. In Example 16-16, you can see the top-level attributes for object server1 using the **show** command.

Example 16-16 Finding top-level attributes for a given object

```
wsadmin>set serv [$AdminConfig getid "/Node:MyServ/Server:server1/"]

server1(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#Server_1)
wsadmin>$AdminConfig show $serv{components
{(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#NameServer_1)
(cells/MyServNetwork/node/MyServ/servers/server1:server.xml#ApplicationServer_1
)}}
{customServices {}}
{errorStreamRedirect
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#StreamRedirect_1)}
{name server1}
{outputStreamRedirect
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#StreamRedirect_2)}
{processDefinition
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#JavaProcessDef_1)}
{services
{(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#PMIService_1)
(cells/MyServNetwork/nodes/
ka0kkwd/servers/server1:server.xml#AdminService_1)
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#Tra
eService_1)
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#RASLoggingService_
1) (cells/MyServNetwor
/nodes/MyServ/servers/server1:server.xml#ObjectRequestBroker_1)}}
{stateManagement
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#StateManageable_1)
}
{statisticsProvider
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#StatisticsProvider
_1)}
```

If you want to see values for a particular attribute, you can use the **showAttribute** command. In Example 16-17, the values for name attribute and the services attribute of server1 are listed.

Example 16-17 Finding values for particular attribute for a given object

```
wsadmin>$AdminConfig showAttribute $serv name
server1

wsadmin>$AdminConfig showAttribute $serv services

{(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#PMIService_1)
(cells/MyServNetwork/nodes/MyServ/s
rvers/server1:server.xml#AdminService_1)
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#TraceService_)
(cells/MyServNetwork/nodes/MyServ/servers/server1:server.xml#RASLoggingService_
1)
(cells/MyServNetwork/nodes/mk0kkwd/servers/server1:server.xml#ObjectRequestBrok
er_1)}
```

Another useful command to list all attributes and their values is **showall** command of AdminConfig object. This command returns all the attributes of a given object.

16.4 Common operational administrative tasks using wsadmin

In this section we describe how wsadmin can be used to carry out common WebSphere operation tasks. The section discusses a general approach for operational tasks and gives specific examples of common administrative tasks.

16.4.1 General approach for operational tasks

In order to invoke an operation on a running MBean, you first need to know the object name of the running object. Then you invoke the method on a fully qualified object name. This means that invoking operations usually involves two type of commands:

- Find the object name
- Invoke the operation

In simple cases, two commands can be combined into one command.

Similarly in order to change an attribute of a running object, you first need to know the object name of that running object. This means that getting or setting attributes involves a sequence of two commands:

- Find the object name of the running object/MBeans
- Get or set attributes for that running object

Note: You can use the queryNames and completeObjectName commands of the AdminControl object to identify the name of a running object. See "Getting help for wsadmin objects" on page 791 for information on how to do this.

16.4.2 Specific examples of common administrative tasks

Common operational tasks performed using wsadmin include:

- Starting and stopping the Deployment Manager
- Starting and stopping nodes
- Starting and stopping application servers
- Starting, stopping, and viewing enterprise applications
- Starting and stopping clusters
- Generating the Web server plug-in configuration file
- Enabling tracing for WebSphere components

Note: Some of the examples used in this section need Network Deployment installed. In our test environment, we installed both WebSphere Application Server and Network Deployment on the same machine. To show the command syntax, we use WebSphere sample applications that get installed on server1 while installing WebSphere.

The elements of our environment include:

- Server node: MyServ
- Deployment Manager node: MyServManager
- Node agent server: nodeagent
- JMS server: jmsserver
- Server: server1

16.4.3 Managing the Deployment Manager

This section describes how to start and stop tasks on the Deployment Manager using the WebSphere scripting interface wsadmin.

Starting the Deployment Manager

wsadmin works on MBeans and since the MBean representing the Deployment Manager is not available unless the process is running, you have to use the **`startManager`** command to start it (see A.1, “startManager” on page 834). On Windows, you also have the option of starting the IBM WebSphere Application Server V5 - dmgr Windows service or by clicking **Start ->Programs**.

Stopping the Deployment Manager

The Deployment Manager can be stopped using the AdminControl object and invoking the stopServer command. To invoke stopServer, you must provide the Deployment Manager name and the node name. Example 16-18 shows an example of stopping Deployment Manager.

Example 16-18 Stopping Deployment Manager using a single line command

```
wsadmin>$AdminControl stopServer dmgr MyServManager

WASX7337I: Invoked stop for server "dmgr" Waiting for stop completion.
WASX7264I: Stop completed for server "dmgr" on node "MyServManager"
```

The stop operation can also be performed by invoking the stop method of the AdminControl object on the MBean representing the Deployment Manager. We need to identify the MBean that represents the Deployment Manager using the queryNames command of AdminControl object.

Example 16-19 shows the command to query the MBeans information and the nested command to stop the Deployment Manager

Example 16-19 Getting MBean information and stopping the Deployment Manager

```
wsadmin>$AdminControl queryNames type=Server,node=MyServManager,*

WebSphere:cell=MyServNetwork,name=dmgr,mbeanIdentifier=server.xml#Server_1,type
=Server,node=MyServManager,process=dmgr,processType=DeploymentManager

wsadmin>$AdminControl invoke [$AdminControl queryNamestype=Server,node=MyServM
anager,*] stop {}
```

16.4.4 Managing nodes

This section describes how to perform common administration tasks on nodes and their node agent using wsadmin.

Starting a node agent

As with the Deployment Manager, the node agent can't be started with wsadmin because there are no MBeans available yet. Use the **startNode** command to start the node agent (see A.3, "startNode" on page 838).

1. Change directory to the bin directory of the base application server installation for that node.
2. Run **startNode** from the command line.

If successful, the node agent server process ID will be displayed on the window. If there are any errors, check the log file for the node agent process:

```
<WAS_HOME>/logs/dmgr/SystemOut.log
```

Stopping a node agent

The node agent process controls all of the WebSphere managed processes on a node. Therefore stopping a node agent limits the ability to issue any further commands against managed servers. In a WebSphere cell, there is one node agent per node.

Node agents can be stopped by invoking the stopServer command of the AdminControl object. The name of the node agent server and the name of the node need to be supplied as arguments. Example 16-20 on page 804 shows the command to stop a node agent.

Example 16-20   Single line command to stop a node agent

```
wsadmin>$AdminControl stopServer nodeagent MyServ

WASX7337I: Invoked stop for server "nodeagent" Waiting for stop completion.
WASX7264I: Stop completed for server "nodeagent" on node "MyServ"
```

The stop operation of the node agent can also be performed by invoking the stop operation on the MBean representing the node agent. We first need to identify the MBean for the node agent using the queryNames command.

Example 16-21 shows the command syntax to query MBean information for the node agent and to invoke the stop method on the identified MBean.

Example 16-21   Getting MBean information for a node agent

```
wsadmin>$AdminControl queryNames type=Server,node=MyServ,name=nodeagent,*

WebSphere:name=nodeagent,process=nodeagent,platform=common,node=MyServ,version=
5.0,type=Server,mbeanIdentifier=cells/MyServNetwork/nodes/MyServ/servers/nodeag
ent/server.xml#Server_1076495775772,cell=MyServNetwork,processType=NodeAgent

wsadmin>$AdminControl invoke [$AdminControl queryNames
type=Server,node=MyServ,name=nodeagent,*] stop {}
```

16.4.5 Managing application servers

This section describes how to perform common administration tasks on application servers using wsadmin.

Starting an application server

In a Network Deployment environment the node agent can start an application server. Example 16-22 shows the command for starting the server1 application server.

Example 16-22   Start an application server

```
wsadmin>$AdminControl startServer server1 MyServ

WASX7262I: Start completed for server "server1" on node "MyServ"
```

Stopping an application server

Example 16-23 on page 805 shows the command for stopping the server1 application server.

Example 16-23 Stop an application server

```
wsadmin>$AdminControl stopServer server1 MyServ
```

You can also use the AdminControl object to invoke the stop method on the application server. To do this you will need to identify the MBean representing the application server. Example 16-24 shows the command to query the MBean information of the application server.

Example 16-24 MBean information for Application Server server1

```
wsadmin>$AdminControl queryNames type=Server,node=MyServ,name=server1,*

WebSphere:name=server1,process=server1,platform=common,node=MyServ,version=
5.0,type=Server, mbeanIdentifier=
cells/MyServNetwork/nodes/MyServ/servers/server1/server.xml#Server_1,cell=MySer
vNetwork,processType=ManagedProcess
```

Example 16-25 shows the nested command syntax to stop the application server.

Example 16-25 Stopping an application server

```
wsadmin>$AdminControl invoke [$AdminControl completeObjectName
type=Server,node=MyServ,name=server1,*] stop {}
```

The nested command in Example 16-25 can be simplified by setting a variable to the object name. Example 16-26 shows how to set a variable to an object name and then invoke the stop method on that variable.

Example 16-26 Set a variable to object name

```
wsadmin>set myserv [$AdminControl completeObjectName
type=Server,node=MyServ,name=server1,*]

wsadmin>$AdminControl invoke $myserv stop {}
```

If there are multiple application servers running on a node, you can stop all the servers from a single script. Example 16-27 on page 806 shows a script that will stop all application servers on the MyServ node. In this example, the node name is hard-coded, but it is also possible to write Jacl code that will accept the node name from the command line or a menu.

The script is invoked from a command line in this way:

```
cd \WebSphere\DeploymentManager\bin
wsadmin -f <script_filename>
```

Example 16-27 Stopping all application servers on a node

```
set servername [$AdminControl queryNames node=MyServ,type=Server,*]
foreach item $servername {
set shortname [$AdminControl getAttribute $item name]
set completename [$AdminControl completeObjectName
type=Server,node=MyServ,name=$shortname,*]
puts "Stopping server : $shortname"
$AdminControl invoke $completename stop {}
}
```

16.4.6 Managing enterprise applications

This section describes how to perform common administration tasks on enterprise applications using the scripting interface, wsadmin.

Viewing installed applications

The $AdminApp list command can be used to list to applications installed under an application server. Example 16-28 shows the use of this command and the output.

Example 16-28 Listing installed applications

```
wsadmin> $AdminApp list

adminconsole
WebbankV5
TechnologySamples
DefaultApplication
petstore
ivtApp
PlantsByWebSphere
MDBSamples
SamplesGallery
filetransfer
```

You can also do this by querying the MBeans for running applications on a node. Example 16-29 on page 807 shows how this is done.

Example 16-29 Listing applications by MBeans query

```
wsadmin>$AdminControl queryNames type=Application,node=MyServ,*

WebSphere:name=DefaultApplication,process=server1,platform=common,node=MyServ,J
2EEName=DefaultApplication,Server=server1,version=5.0,type=Application,mbeanIde
ntifier=cells/MyServNetwork/applications/DefaultApplication.ear/deployments/Def
aultApplication/deployment.xml#ApplicationDeployment_1076417745705,cell=MyServN
etwork.......
```

Running objects are represented by MBeans. If an object is not running the MBean for that object does not exist. Based on this, we can write a simple Jacl script that will display running applications.

Example 16-30 shows a script that will list the installed applications using the AdminApp object. The data obtained is configurational data and can't be interrogated to determine runtime status. So queryNames is used for each installed application to see if MBean exists (the application is running). If the application is running, queryNames returns a name. Otherwise it returns a null value.

Example 16-30 Script to display the status of Applications

```
set application [$AdminApp list]
foreach app $application {
   set objName [$AdminControl queryNames type=Application,name=$app,*]
   if {[ llength $objName] ==0} {
   puts "The Application $app is not running"
   } else {
   puts "The Application $app is running"
   }
    }
```

Stopping a running application

To stop a running application, we use the AdminControl object and invoke the stopApplication method on the MBean of the running application. Example 16-31 shows the sequence of commands used to query the MBean and to stop the application.

Example 16-31 Stopping a running application

```
wsadmin> set appservername [$AdminControl queryNames
type=ApplicationManager,node=MyServ,process=server1,*]

wsadmin>$AdminControl invoke $appservername stopApplication DefaultApplication
```

Starting a stopped application

To start a stopped application, we use the AdminControl object and invoke the startApplication method on the stopped application. This requires the identity of the application server MBean. Example 16-32 shows the sequence of commands used to start the DefaultApplication application.

Example 16-32 Starting a stopped application

```
wsadmin> set appservername [$AdminControl queryNames
type=ApplicationManager,node=MyServ,process=server1,*]

wsadmin>$AdminControl invoke $appservername startApplication DefaultApplication
```

16.4.7 Managing clusters

This section describes how to perform common administration tasks on clusters using wsadmin.

Starting a cluster

Example 16-33 shows the sequence of commands needed to start a cluster. The first command lists the configured clusters in the cell. In this case, there is only one cluster, testCluster. The second command initializes a variable called “clid” with the cluster object name. The final command invokes the start method on the cluster object.

Example 16-33 Start a cluster

```
wsadmin>$AdminControl queryNames type=Cluster,*

WebSphere:cell=MyServNetwork,name=testCluster,mbeanIdentifier=testCluster,type=
Cluster,node=MyServManager,process=dmgr

wsadmin>set clid [$AdminControl completeObjectName
type=Cluster,name=testCluster,*]

WebSphere:cell=MyServNetwork,name=testCluster,mbeanIdentifier=testCluster,type=
Cluster,node=MyServManager,process=dmgr

wsadmin>$AdminControl invoke $clid start
```

Stopping a cluster

Example 16-34 on page 809 shows the sequence of commands used to stop a cluster. The first command lists the configured clusters in the cell. The second command initializes a variable called “clid” with the cluster object name. The final command invokes the stop method on the cluster object.

Example 16-34 Stopping a cluster

```
wsadmin>$AdminControl queryNames type=Cluster,*

WebSphere:cell=MyServNetwork,name=testCluster,mbeanIdentifier=testCluster,type=
Cluster,node=MyServManager,process=dmgr

wsadmin>set clid [$AdminControl completeObjectName
type=Cluster,name=testCluster,*]

WebSphere:cell=MyServNetwork,name=testCluster,mbeanIdentifier=testCluster,type=
Cluster,node=MyServManager,process=dmgr

wsadmin>$AdminControl invoke $clid stop
```

16.4.8 Generating the Web server plug-in configuration

Example 16-35 shows the sequence of commands used to regenerate the Web server plug-in configuration file. The first command identifies the MBean for the Web server plug-in on a node. The second command generates the Web server plug-in.

Example 16-35 Generating the Web server plug-in

```
wsadmin>set pluginGen [$AdminControl completeObjectName
type=PluginCfgGenerator,*]

wsadmin>$AdminControl invoke $pluginGen generate
"c:/WebSphere/DeploymentManager c:/WebSphere/DeploymentManager/config
MyServNetwork null null plugin-cfg.xml"
```

The argument for the **`generate`** command includes:

- Install root directory
- Configuration root directory
- Cell name
- Node name
- Server name
- Output file name

You can use `null` as an argument. In this example, both node and server are set to `null`. The **`generate`** operation generates a plug-in configuration for all the nodes and servers residing in the cell. The output file, plugin-cfg.xml, is created in the configuration root directory.

16.4.9 Enabling tracing for WebSphere component

Tracing used for problem determination is discussed in 15.4, “Traces” on page 727. This section illustrates how to enable tracing for a server process using the **`setAttribute`** command on the TraceService MBean.

In a Network Deployment environment there are multiple server processes and therefore multiple TraceService MBeans. Example 16-36 shows how to use queryNames to list the TraceService MBeans.

Example 16-36 List of TraceService MBeans

```
wsadmin>$AdminControl queryNames type=TraceService,*

WebSphere:platform=common,cell=MyServNetwork,version=5.0,name=TraceService,mbea
nIdentifier=cells/MyServNetwork/nodes/MyServ/servers/WebbankServer/server.xml#T
raceService_1,type=TraceService,node=MyServ,process=WebbankServer

WebSphere:platform=common,cell=MyServNetwork,version=5.0,name=TraceService,mbea
nIdentifier=cells/MyServNetwork/nodes/MyServ/servers/jmsserver/server.xml#Trace
Service_1076495775833,type=TraceService,node=MyServ,process=jmsserver

WebSphere:platform=common,cell=MyServNetwork,version=5.0,name=TraceService,mbea
nIdentifier=cells/MyServNetwork/nodes/MyServ/servers/nodeagent/server.xml#Trace
Service_1076495775772,type=TraceService,node=MyServ,process=nodeagent

WebSphere:platform=common,cell=MyServNetwork,version=5.0,name=TraceService,mbea
nIdentifier=cells/MyServNetwork/nodes/MyServManager/servers/dmgr/server.xml#Tra
ceService_1,type=TraceService,node=MyServManager,process=dmgr
```

To start tracing for a server, you need to locate the TraceService MBean for the server process using the **`completeObject`** command. Example 16-37 shows how to do this, using a variable called “ts” which is set to the value of the tracing service MBean. In the second step the **`setAttribute`** command is used to enable the tracing.

Example 16-37 Enable tracing using TraceService mbean

```
wsadmin>set ts [$AdminControl completeObjectName
type=TraceService,process=WebbankServer,*]

wsadmin>$AdminControl setAttribute $ts traceSpecification
com.ibm.ejs.*=all=enabled
```

16.5 Common configuration tasks

In this section we describe how wsadmin can be used to create, modify, and change WebSphere Application Server configuration. The section is described in two parts as follows:

- General approach for configuration tasks
- Specific examples of configuration tasks

16.5.1 General approach for configuration tasks

The are many possible configuration tasks that can be performed in a WebSphere environment. Rather than document every possible modification, we describe a general approach to use when performing configuration tasks and then give a few specific examples.

This general approach has three steps:

- Find the object you want to change using $AdminConfig **`getid`**.
- Make the change or create the configuration using $AdminConfig **`modify`** or **`create`**.
- Save the changes using $AdminConfig **`save.`**

The **`create`** and **`modify`** commands use an attribute list. In general, the attribute is supplied as a list of Jacl lists. A Jacl list can be constructed using name and value pair as follows:

```
{ { name1 value1} {name2 value2} {name3 value3}.........}
```

The attributes for a WebSphere configuration object are often deeply nested. If you need to modify a nested attribute you can get the ID of the object and modify it directly. This is the preferred method, although it requires more lines of scripting.

16.5.2 Specific examples of WebSphere configuration tasks

This section describes how a variety of typical configuration tasks can be done using wsadmin>set tasks, including:

- Application server
  - Create/remove an application server
- Enterprise application
  - Install/uninstall an enterprise application
  - Change attributes of an enterprise application

- Configure and modify WebSphere configuration
  - Configure virtual hosts
  - Configure JDBC providers
  - Edit an application server
  - Create a cluster
  - Add member to a cluster

Creating an application server

The general syntax for creating an application server is:

```
$adminConfig create <type> <node configuration id> <attribute list>
```

Example 16-38 illustrates creating an application server. The first command initializes a variable called “node” to set the value of the node configuration ID. The second command creates the server on the node.

Example 16-38 Creating an application server

```
wsadmin>set node [$AdminConfig getid /Node:MyServ/]
wsadmin>$AdminConfig create Server $node {{name testserver}}
```

Removing an application server

The general syntax for removing an application server is:

```
$AdminConfig remove <server Config id>
```

Example 16-39 illustrates removing an application server. The first command sets the configuration ID of the application server as a variable. The second command removes the server.

Example 16-39 Remove an application server

```
wsadmin>set server [$AdminConfig getid /Node:MyServ/Server:testserver/]
wsadmin>$AdminConfig remove $server
```

Installing an enterprise application

There are two options for installing an application:

- Interactive installation using the **`installInteractive`** command. The interactive install will prompt you for options. The syntax is:

  ```
  $AdminApp installInteractive <ear_file_location>
  ```

  For example:

  ```
  $AdminApp installInteractive
  c:\\websphere\\deploymentmanager\\installableapps\\WebbankV51.ear
  ```

(Note that in Windows, you will need to use either forward slashes (/) or double backslashes (\\) when specifying the path to the .ear file.

- Non-interactive installation using the **`install`** command.

Using the install command

The general syntax for installing an enterprise application is as follows:

```
$AdminApp install <location of the ear file> { task or non-task option}
```

There are two types of options that can be specified when you use the **`install`** command:

- Install task options. To see a list of these options, use the following syntax:

  ```
  $AdminApp options
  ```

  The list includes options that allow you to specify options for the installation such as server name, cluster name, install directory, etc.

- Application-specific options. To see a list of these options, use the following syntax:

  ```
  wsadmin>$AdminApp options <ear_file_location>
  ```

  For example:

  ```
  wsadmin>$AdminApp options c:/temp/PerfServletApp.ear
  ```

  The list of options includes things that define application information, security role mapping, module to virtual host mapping, and whether to pre-compile JSPs.

Note: All options supplied for the **`install`** command must be supplied in a single string. In Jacl, a single string is formed by collecting all options within curly braces or double quotes:

```
$AdminApp install c:/temp/application.ear { -server serv2 -appname -TestApp}
```

Example 16-40 shows an example of installing a new application called TestApp on a server called testserver.

Example 16-40 Installing an application

```
wsadmin>$AdminApp install c:/temp/Deployed_PerfServletApp.ear {-server
testserver -appname TestApp}
wsadmin>$AdminConfig save
```

Uninstalling an enterprise application

The general syntax for uninstalling enterprise application is as follows:

```
$AdminApp uninstall <name of the application>
```

Example 16-41 shows an example of uninstalling an application.

Example 16-41 Uninstalling an enterprise application

```
wsadmin>$AdminApp uninstall TestApp

ADMA5017I: Uninstallation of TestApp started.
ADMA5102I: Deletion of config data for TestApp from config repository completed
successfully.
ADMA5106I: Application TestApp uninstalled successfully.

wsadmin>$AdminConfig save
```

Editing an enterprise application

Editing of an enterprise application can be done either interactively or non-interactively. The following commands are available for editing:

- Interactively using the **editInteractive** command. This will prompt you for input. The syntax is:

  ```
  $AdminApp editInteractive app1
  ```

- Non-interactively using the **edit** command

Using the edit command

The general syntax for editing an enterprise application in non-interactive mode is:

```
$AdminApp edit <application_name> {-taskname {{item1a item2a item3a}
{item1b item2b item3b}.......}}
```

In Example 16-42, you can see how to change the module to server mapping for an application. The options are the same as those you would use during installation with the **install** command.

Example 16-42 Edit an enterprise application

```
wsadmin>$AdminApp edit TestApp { -MapModulesToServers {{perfservlet
perfServletApp.war,WEB-INF/web.xml

WebSphere:cell=MyServNetwork,node=MyServ,server=server1}}}

wsadmin>$AdminConfig save
```

Creating a virtual host

The command to create a virtual host is:

```
$AdminConfig create VirtualHost $cl {name test_host}
```

First, you will need to find the ID of the object you want to change. Virtual host is a WebSphere resource defined in a WebSphere cell. Therefore by creating a virtual host we are modifying the configuration of the WebSphere cell object. We therefore need the ID for our WebSphere cell.

Example 16-43 shows the command syntax to use to get the ID of the cell. It uses a variable called “cl” to hold the object ID of the WebSphere cell.

Example 16-43 Find an object using AdminConfig command

```
wsadmin>set cl [$AdminConfig getid /Cell:MyServNetwork/]

MyServNetwork(cells/MyServNetwork:cell.xml#Cell_1)
```

Once you have the object ID, you can make the necessary changes. Example 16-44 shows the command syntax to create the virtual host and save the configuration.

Example 16-44 Create a virtual host called test_host

```
wsadmin>$AdminConfig create VirtualHost $cl {name test_host}

wsadmin>$AdminConfig save
```

Modifying a virtual host

You can modify a virtual host using the $AdminConfig **modify** command.

Example 16-45 shows an example of modifying a virtual host. The example gets the ID of the test_host virtual host, then uses that ID in the **modify** command to redefine the list of aliases.

Example 16-45 Modifying a virtual host

```
wsadmin>set vh [$AdminConfig getid /VirtualHost:test_host/]

test_host(cells/MyServNetwork:virtualhosts.xml#VirtualHost_3)

wsadmin>$AdminConfig modify $vh {{aliases {{{hostname *} {port 9080}}
{{hostname *} {port 80}} {{hostname *} {port 9081}}}}}

wsadmin>$AdminConfig save
```

Configuring JDBC providers

Example 16-46 on page 816 shows a common method for creating a JDBC provider. The provider is created based on a template.

Using templates: A group of templates are supplied with the WebSphere installation as XML files in the <WAS_HOME>/config/template directory. Within each XML file you will find multiple entries. To use a template, you specify the XML file and the entry within the file that you want to use.

Using templates is especially useful when using wsadmin for configuration purposes. The template reduces the amount of typed input required, speeding up the process and reducing the probability of syntax errors.

The **`listTemplates`** command of the AdminConfig object prints a list of templates matching a given type. These templates may be used with the **`createUsingTemplate`** command.

The JDBC provider will be added at the node scope, so the first command gets the configuration ID for the node and sets a variable called "node" to hold the ID. The second command uses listTemplates to set the temp1 variable to the template ID. The third command creates the JDBC provider using the template.

Example 16-46 Configuring a JDBC driver

```
wsadmin>set node [$AdminConfig getid /Node:MyServ/]

wsadmin>set temp1 [$AdminConfig listTemplates JDBCProvider "DB2 JDBC Provider"]

wsadmin>$AdminConfig createUsingTemplate JDBCProvider $node {{name testdriver}}
$temp1
```

Modifying an application server

Updates made to a server configuration are done using $AdminConfig modify. don't take affect until the server is stopped and started.

Example 16-47 on page 817 shows an example of changing the ping interval for a server called "testserver". Note that it takes several steps to modify the server:

- The first command stops the server.
- A variable is set to the object ID for the server.
- A variable called "pdef" is set with the object ID of the embedded object ProcessDef for testserver. Note that the **`list`** command is used to get the object ID. This is because the ProcessDef object has nested list attributes.
- A variable called "mp" is set with the object ID for the MonitoringPolicy embedded object.
- The server's ping interval is modified using the MonitoringPolicy object ID as the parent object.

- The **save** command saves the change.
- The application server is restarted to effect the change.

Tip: To find the parent object you can run **`$AdminConfig showall <Object id of the application server>`** and then analyze the output to understand the relationship among objects.

You can get a rough idea of the parent/child relationship between objects from the administrative console. The layout of the navigation path follows a logical progression from parent to child. For example, to change the PingInterval you would need to traverse through the following:

Application Server -> Process Definition -> Monitoring Policy -> Ping Interval

Example 16-47 Modifying an application server

```
wsadmin>$AdminControl stopServer testserver MyServ

wsadmin>set s1 [$AdminConfig getid /Node:MyServ/Server:testserver/]

wsadmin>set pdef [$AdminConfig list ProcessDef $s1]

wsadmin>$AdminConfig mp [$AdminConfig list MonitoringPolicy $pdef]

wsadmin>$AdminConfig modify $mp {{pingInterval 120}}

wsadmin>$AdminConfig save

wsadmin>$AdminControl startServer testserver MyServ
```

Creating a cluster

To create a new cluster you need the configuration ID of an existing server and the name to use for the new cluster.

Example 16-48 shows how to create a new cluster called “testCluster” using an existing server called “testserver”.

Example 16-48 Create a ServerCluster object

```
wsadmin>set serverid [$AdminConfig getid /Server:testserver/]

wsadmin>$AdminConfig convertToCluster $serverid testCluster

wsadmin>$AdminConfig save
```

Adding a member to an existing cluster

The createClusterMember is used to create a new server and add it to a cluster.

Example 16-49 shows how to create a new server (testserver2) and make it a member of a cluster (testCluster).

Example 16-49 Add a cluster member

```
wsadmin>set testcluster [$AdminConfig getid /ServerCluster:testCluster/]

wsadmin>set nodeid [$AdminConfig getid /Node:MyServ/]

wsadmin>$AdminConfig createClusterMember $testcluster $nodeid {{memberName
testserver2}}

wsadmin>$AdminConfig save
```

Note: There is no command to remove a server from a cluster. If you no longer wish your server object to be part of the cluster, you should remove the server object.

16.6 Case study: configuring and managing the Webbank application using wsadmin

In this section, we take you through the process we used to set up the environment for our Webbank application and to install it.

16.6.1 Assumptions

You can assume that these tasks have already been performed:

- The Web server has been configured to serve the Webbank application on www.webbank.itso.ibm.com port 90.
- WebSphere Application Server and Network Deployment are installed and configured on the same machine.
- The DB2UNIVERSAL_JDBC_DRIVER_PATH variable has been set to `c:\sqllib\java`.
- The WEBBANK database has been created and populated on IBM DB2. The database is local to the application and no user ID and password are needed to access the database. DB2 is running.

- The Webbank application EAR file is available in the <WAS_HOME>\installableApps directory.
- We are using a Windows platform. The steps would be the same for a UNIX system. The only modification would be in file path designations.

16.6.2 Creating a new virtual host

Example 16-50 shows the command to create a new virtual host called webbank_vhost.

Example 16-50 Creating the virtual host webbank_vhost

```
wsadmin>set cl [$AdminConfig getid /Cell:MyServNetwork/]

wsadmin>$AdminConfig create VirtualHost $cl {{name webbank_vhost}}

wsadmin>$AdminConfig save
```

Example 16-51 shows the command to add the host name and port information to the new virtual host.

Example 16-51 Modify the webbank_vhost alias information

```
wsadmin>set vh [$AdminConfig getid /VirtualHost:webbank_vhost/]

wsadmin>$AdminConfig modify $vh {{aliases {{{hostname www.webbank.itso.ibm.com}
{port 90}}}}}

wsadmin>$AdminConfig save
```

With these settings the application server will accept URLs connecting to host name www.webbank.itso.ibm.com on port 90. You do not need to specify the MIME table, since it will be created by default.

16.6.3 Creating the JDBC provider

Example 16-52 on page 820 shows the commands needed to create and configure the JDBC driver for a DB2 database. It uses the templates provided with WebSphere Application Server. You can see the definitions of these templates in <WAS_ND_HOME>/config/templates/system/jdbc-resource-provider-templates.xml.

Example 16-52 Creating the JDBC provider

```
wsadmin>set node [$AdminConfig getid /Node:MyServ/]

wsadmin>$AdminConfig listTemplates JDBCProvider "DB2 Universal JDBC Driver
Provider"

"DB2 Universal JDBC Driver Provider XA)
(templates/system:jdbc-resource-provider-templates.xml#JDBCProvider_DB2_UNI_2)"
"DB2 Universal JDBC Driver Provider
(templates/system:jdbc-resource-provider-templates.xml#JDBCProvider_DB2_UNI_1)"

wsadmin>set temp1 [lindex [$AdminConfig listTemplates JDBCProvider "DB2
Universal JDBC Driver Provider"] 1]

DB2 Universal JDBC Driver Provider
(templates/system:jdbc-resource-provider-templates.xml#JDBCProvider_DB2_UNI_1)

wsadmin>$AdminConfig createUsingTemplate JDBCProvider $node {{name
WebbankUDBDriver}} $temp1

wsadmin>$AdminConfig save
```

Note: `lindex` is a Jacl command. It retrieves an element from a list. This command treats the list as a Tcl list and returns the indexed element from it. For example, "0" refers to the first element of the list. For further information about Jacl commands, check the following URL:

http://www.javasim.org/ref.script/tcljacl.htm

16.6.4 Creating a data source

To create a data source for the database using the JDBC provider defined earlier, we will do the following:

- Create data source using a template.
- Modify the jndiName attribute for the data source.
- Modify the databaseName attribute.
- Create a CMPConnectorFactory to make the data source CMP enabled.
- Test connectivity using the created data source.

Example 16-53 on page 821 shows the command sequence to create a data source. As you can see in the example, first we get the configuration ID of the JDBC provider created earlier. Then we list all the available templates that can be used for creating data sources using the `listTemplates` command (you can see the templates by browsing the file):

`<WAS_ND_HOME>/config/templates/system/jdbc-resource-provider-templates.xml.`

A specific template is selected using the **lindex** command. The final command creates the data source.

Example 16-53 Creating a data source using a template

```
wsadmin>set dbprovider [$AdminConfig getid
/Node:MyServ/JDBCProvider:WebbankUDBDriver/]

WebbankUDBDriver(cells/MyServNetwork/nodes/MyServ:resources.xml#JDBCProvider_10
77139054359)

wsadmin>$AdminConfig listTemplates DataSource

(templates/system:jdbc-resource-provider-templates.xml#DataSource_1)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_2)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_3)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_3_2)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_4)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_4_2)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DB2_RRS)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DB2_UNI_1)
(templates/system:jdbc-resource-provider-templates.xml#DataSource_DB2_UNI_2)
...

wsadmin>set temp2 [lindex [$AdminConfig listTemplates DataSource] 7]

(templates/system:jdbc-resource-provider-templates.xml#DataSource_DB2_UNI_1)

wsadmin>$AdminConfig createUsingTemplate DataSource $dbprovider {{name
webbankds}} $temp2

webbankDS(cells/MyServNetwork/nodes/MyServ:resources.xml#DataSource_10771421710
00)

wsadmin>$AdminConfig save
```

Now that the data source has been created, we modify it to set the JNDI name for the data source to `webbankDS`. Example 16-54 on page 822 shows how we get the configuration ID for the JDBC provider and then modify the jndiName attribute.

Example 16-54 Set jndiiName for data source

```
wsadmin>set datasource [$AdminConfig getid /DataSource:webbankDS/]

webbankDS(cells/MyServNetwork/nodes/MyServ:resources.xml#DataSource_10771423402
19)

wsadmin>$AdminConfig modify $datasource {{jndiName webbank/jdbc/webbank}}

wsadmin>$AdminConfig save
```

The next modification necessary is to set the values of the custom properties required to connect to the database. In this example, the properties required are the database name (webbank) and the server name (MyServ). These attributes are nested attributes, so we need to identify where they are nested.

First, we get the configuration ID for the data source. Then we list the properties to find the location of the databaseName and serverName attributes. From the output of the **showall** command, you can see that they are nested within the resourceProperties attribute. Once you identify the location of each attribute, you can modify them using the **modify** command as shown in Example 16-55.

Example 16-55 Modify the databaseName attribute for the data source

```
wsadmin>set datasource [$AdminConfig getid /DataSource:webbankDS/]
webbankDS(cells/MyServNetwork/nodes/MyServ:resources.xml#DataSource_10771423402
19)

wsadmin>set ps [$AdminConfig showAttribute $datasource propertySet]
(cells/MyServNetwork/nodes/MyServ:resources.xml#J2EEResourcePropertySet_1077142
340219)

wsadmin>set rps [lindex [$AdminConfig showAttribute $ps resourceProperties] 0]

databaseName(cells/MyServNetwork/nodes/MyServ:resources.xml#J2EEResourcePropert
y_1077142340219)
driverType(cells/MyServNetwork/nodes/MyServ:resources.xml#J2EEResourceProperty_
1077142340220)
serverName(cells/MyServNetwork/nodes/MyServ:resources.xml#J2EEResourceProperty_
1077142340221)
....
```

```
wsadmin>foreach rp $rps {if {[regexp databaseName $rp] == 1} {$AdminConfig
modify $rp {{value webbank}}}}

wsadmin>foreach rp $rps {if {[regexp serverName $rp] == 1} {$AdminConfig modify
$rp {{value MyServ}}}}

wsadmin>$AdminConfig save
```

Last, we need to CMP-enable the data source. To do this, we create a CMPConnectorFactory for the given data source.

First, we need to find the parent object type of CMPConnectorFactory. We use the **parent** command to discover that J2CResourceAdapter is the parent object type for CMPConnectorFactory.

Next, we set the scope to the node level by selecting the J2CResourceAdapter for the node from the list.

Finally, we use the **create** command to create the CMPConnectorFactory and save the configuration.

Example 16-56 Creating CMPConnectorFactory for a data source

```
wsadmin>$AdminConfig parents CMPConnectorFactory

J2CResourceAdapter

wsadmin>$AdminConfig list J2CResourceAdapter

"WebSphere Relational Resource Adapter
(cells/MyServNetwork/nodes/MyServ/servers/server1:resources.xml#builtin_rra)"
"WebSphere Relational Resource Adapter
(cells/MyServNetwork/nodes/MyServ:resources.xml#builtin_rra)"
"WebSphere Relational Resource Adapter
(cells/MyServNetwork/nodes/MyServManager:resources.xml#builtin_rra)"
"WebSphere Relational Resource Adapter
(cells/MyServNetwork:resources.xml#builtin_rra)"

wsadmin>set rsadapter [lindex [$AdminConfig list J2CResourceAdapter] 1]

WebSphere Relational Resource Adapter
(cells/MyServNetwork/nodes/MyServ:resources.xml#builtin_rra)

wsadmin>$AdminConfig getid /DataSource:webbankDS/

webbankDS(cells/MyServNetwork/nodes/MyServ:resources.xml#DataSource_10771423402
19)
```

```
wsadmin>$AdminConfig create CMPConnectorFactory $rsadapter {{name webbankCMP}
{cmpDatasource webbankDS
(cells/MyServNetwork/nodes/MyServ:resources.xml#DataSource_1077142340219)}}

webbankCMP(cells/MyServNetwork/nodes/MyServ:resources.xml#CMPConnectorFactory_1
077192098266)

wsadmin>$AdminConfig save
```

Note: When the connection factory is created, a JNDI name is assigned to it. The JNDI name will be in the format: eis/<datasource_name>_CMP.

Last, we test the data source using the `testConnection` command. The test can be seen in Example 16-57.

Example 16-57 Testing a data source

```
wsadmin>set datasource [$AdminConfig getid /DataSource:webbankDS/]

webbankDS(cells/MyServNetwork/nodes/MyServ:resources.xml#DataSource_10771423402
19)

wsadmin>$AdminControl testConnection $datasource

SRA8025I: Successfully connected to DataSource.
```

16.6.5 Creating the application server

We decided to create a new application server to run the Webbank application. Example 16-58 shows the commands we used to create a new server called webbankas.

Example 16-58 Create application server WebbankAS

```
wsadmin>set node [$AdminConfig getid /Node:MyServ/]

MyServ(cells/MyServNetwork/nodes/MyServ:node.xml#Node_1)

wsadmin>$AdminConfig create Server $node {{name WebbankServer}}

WebbankServer(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xml
#Server_1077196896969)

wsadmin>$AdminConfig save
```

The following section shows how to modify the new application server. Normally all of the following attributes are supplied with the **create** command. We do it this way for demonstration purposes.

Changing the server JVM properties

To change the values for JVM properties, we need to locate the attribute. Example 16-59 shows the commands we used to locate the nested attributes for the JVM properties and to modify the initial heap size and garbage collection properties.

Example 16-59 Locate nested attribute for Java virtual machine properties

```
wsadmin>set as [$AdminConfig getid /Node:MyServ/Server:WebbankServer/]

WebbankServer(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xml
#Server_1077196896969)

wsadmin>$AdminConfig show $as

wsadmin>set pd [$AdminConfig list ProcessDef $as]

(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xml#JavaProcessD
ef_1077196896984)

wsadmin>$AdminConfig show $pd

{environment {}}
....
{jvmEntries {(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xm
l#JavaVirtualMachine_1077196896984)}}
...

wsadmin>set jvmattr [$AdminConfig list JavaVirtualMachine $pd]

(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xml#JavaVirtualM
achine_1077196896984)
```

```
wsadmin>$AdminConfig show $jvmattr

{bootClasspath {}}
{classpath {}}
{debugArgs "-Djava.compiler=NONE -Xdebug -Xnoagent
-Xrunjdwp:transport=dt_socket
,server=y,suspend=n,address=7777"}
{debugMode false}
{disableJIT false}
{genericJvmArguments {}}
{hprofArguments {}}
{initialHeapSize 50}
{maximumHeapSize 256}
{runHProf false}
{systemProperties {}}
{verboseModeClass false}
{verboseModeGarbageCollection false}
{verboseModeJNI false}
```

Example 16-60 shows the commands used to change the intialHeapSize and verboseModeGarbageCollection attributes for the WebbankServer application server.

Example 16-60 Changing JVM properties

```
wsadmin>$AdminConfig modify $jvmattr {{initialHeapSize 128}}

wsadmin>$AdminConfig modify $jvmattr {{verboseModeGarbageCollection true}}

wsadmin>$AdminConfig show $jvmattr

...
{initialHeapSize 128}
{maximumHeapSize 256}
{verboseModeGarbageCollection true}
...
wsadmin>$AdminConfig save
```

Changing the HTTP transport port

To change the default Web container port of the webbankas application server, we retrieved the ID of the server's Web container and then modified the "transports" attribute. Example 16-61 on page 827 shows the commands used to change the HTTP transport of the Web container.

Example 16-61 Changing the HTTP transport of the application server

```
wsadmin>set as [$AdminConfig getid /Node:MyServ/Server:WebbankServer/]

WebbankServer(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xml
#Server_1077196896969)

wsadmin>set webcontainer [$AdminConfig list WebContainer $as]

(cells/MyServNetwork/nodes/MyServ/servers/WebbankServer:server.xml#WebContainer
_1077196896984)

wsadmin>$AdminConfig modify $webcontainer {{transports:HTTPTransport
{{{sslEnabled true} {sslConfig DefaultSSLSettings} {address {{host *} {port
9085}}}}}}}

wsadmin>$AdminConfig save
```

16.6.6 Installing the Webbank enterprise application

To install the Webbank application, we need to specify two options on the install command to specify the data source and virtual host created for the application:

- DataSourceFor20CMPBeans: Specifies the data source for 2.x CMP beans.
- MapWebModToVH: Specifies the virtual host to use for the Web modules.

Changing the data source binding

The DataSourceFor20CMPBeans task allows you to specify a data source for the EJBs in an EAR file, overriding the default binding. You can use the **`taskInfo`** command to display information on the command and to see the default bindings set in the EAR file. The output of the **`taskInfo`** command can be seen in Example 16-62.

Example 16-62 Find information of DataSourceFor20CMPBeans for a EAR file

```
wsadmin>$AdminApp taskInfo
c:/WebSphere/DeploymentManager/installableApps/WebbankV51.ear
DataSourceFor20CMPBeans

DataSourceFor20CMPBeans: Specifying Data Sources for Individual 2.x CMP Beans

Specify an optional data source for each 2.x CMP bean. Mapping a specific data
source to a CMP bean will override the default data source for the module
containing the Enterprise bean.

WASX7348I: Each element of the DataSourceFor20CMPBeans task consists of the
following 5 fields: "EJBModule", "EJB", "uri", "JNDI", "resAuth".
```

```
Of these fields, the following may be assigned values: "JNDI" "resAuth"
and the following are required:

The current contents of the task after running default bindings are:
EJBModule: webbankEntityEJBs
EJB: CustomerAccount
uri: webbankEntityEJBs.jar,META-INF/ejb-jar.xml
JNDI: null
resAuth: cmpBinding.perConnectionFactory

EJBModule: webbankEntityEJBs
EJB: BranchAccount
uri: webbankEntityEJBs.jar,META-INF/ejb-jar.xml
JNDI: null
resAuth: cmpBinding.perConnectionFactory

EJBModule: webbankEntityEJBs
EJB: Customer
uri: webbankEntityEJBs.jar,META-INF/ejb-jar.xml
JNDI: null
resAuth: cmpBinding.perConnectionFactory
```

When you specify that a data source will be used for CMP, a connection factory is created for the WebSphere Relational Resource Adapter resource adapter. In our case, the JNDI name of this connection factory is eis/webbank/jdbc/webbank_CMP. You can see from the output of the **`taskInfo`** command that the default JNDI names are currently `null`. We needed to define the JNDI name to match the connection factory JNDI name.

Example 16-63 on page 829 shows how we used variables to contain the new attribute list for each EJB and then bundled them into one list using the Jacl **`list`** command. The variable created here will be used during the application install.

Example 16-63 Building an attribute list for data source

```
wsadmin>set ds1 [list webbankEntityEJBs BranchAccount
webbankEntityEJBs.jar,META-INF/ejb-jar.xml eis/webbank/jdbc/webbank_CMP
cmpBinding.perConnectionFactory]

wsadmin>set ds2 [list webbankEntityEJBs CustomerAccount
webbankEntityEJBs.jar,META-INF/ejb-jar.xml eis/webbank/jdbc/webbank_CMP
cmpBinding.perConnectionFactory]

wsadmin>set ds3 [list webbankEntityEJBs Customer
webbankEntityEJBs.jar,META-INF/ejb-jar.xml eis/webbank/jdbc/webbank_CMP
cmpBinding.perConnectionFactory]

wsadmin>set ds [list $ds1 $ds2 $ds3]
```

Changing the virtual host binding

The MapWebModToVH task allows you to bind a virtual host to a WAR module, overriding the setting in the EAR file. You can use the **`taskInfo`** command to see information on the command and to see the default bindings set in the EAR file. The output of the **`taskInfo`** command can be seen in Example 16-64.

Example 16-64 Finding information of Virtual host in WebBank EAR file

```
wsadmin>$AdminApp taskInfo
c:/WebSphere/DeploymentManager/installableApps/WebbankV51.ear MapWebModToVH

MapWebModToVH: Selecting Virtual Hosts for Web Modules

Specify the virtual host where you want to install the Web modules contained in
your application. Web modules can be installed on the same virtual host or
dispersed among several hosts.

WASX7348I: Each element of the MapWebModToVH task consists of the following 3
fields: "webModule", "uri", "virtualHost". Of these fields, the following may
be assigned values: "virtualHost" and the following are required: "virtualHost"

The current contents of the task after running default bindings are:
webModule: webbankWeb
uri: webbankWeb.war,WEB-INF/web.xml
virtualHost: default_host
```

You can see from the output that WebbankV51.ear has one WAR module, webbankWeb, and that it is bound to the default_host virtual host. We want to change this to Webbank_vhost.

In Example 16-65, you can see the sequence of **`list`** commands used to create another attribute list to be used with the MapWebModToVH task option during installation of the application.

Example 16-65 Building attribute list for MapWebModToVH

```
wsadmin>set vh1 [list webbankWeb webbankWeb.war,WEB-INF/web.xml webbank_vhost]

wsadmin>set vh [list $vh1]
```

Creating the attribute list and installing the application

Finally, we installed the application. First, we combined the two attribute lists created for the data source ($ds) and virtual host ($vh). Then the application was installed with the attributes.

Example 16-66 Creating combined attribute list and installing the application

```
wsadmin>set attrs [list -DataSourceFor20CMPBeans $ds -MapWebModToVH $vh -node 
MyServ -server WebbankServer -appname WebbankApp]

wsadmin>$AdminApp install 
c:/WebSphere/DeploymentManager/installableApps/WebbankV51.ear $attrs
```

Part 6

Appendixes

A

Command-line tools

In this appendix we provide a detailed summary of the command-line tools included with IBM WebSphere Application Server Network Deployment. For each command we include:

- Description
- Important notes and tips
- Syntax
- Arguments
- Examples of usage

A.1 startManager

The **`startManager`** command reads the configuration file for the Deployment Manager process and constructs a launch command for it. Depending on the options specified, the command launches a new Java virtual machine (JVM) to run the manager process, or writes the launch command to a file.

A.1.1 Notes

1. The command must be run from the bin directory of the Deployment Manager installation root: <WAS_ND_HOME>/bin. For example:

   /usr/WebSphere/DeploymentManager/bin

2. The command logs to the files shown in Table A-1.

Table A-1 startManager logs

Log file	Content
<WAS_ND_HOME>/logs/dmgr/SystemOut.log	Console messages
<WAS_ND_HOME>/logs/dmgr/SystemErr.log	Console errors
<WAS_ND_HOME>/logs/dmgr/startServer.log	Command output including trace and errors

3. The Deployment Manager process ID is written to:

   <WAS_ND_HOME>/logs/dmgr/dmgr.pid

4. The Deployment Manager process on startup reads the serverindex.xml file of each node managed by the cell. The host names and IP port numbers are used to discover any running node agent processes.
5. JMX is not used to start up a Deployment Manager. As such, the command does not provide the following arguments: conntype, host, port, user name, or password.

A.1.2 Syntax

The syntax of the **`startManager`** command is:

```
startManager.bat(sh) [options]
```

All arguments are optional. See Table A-2 on page 835.

Table A-2 Options for startManager

Option	Description
`-nowait`	Tells the command not to wait for successful initialization of the launched Deployment Manager process.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to startServer.log.
`-script [<script filename>]`	Generate a launch script instead of starting the server. The script file name is optional. If the file name is not provided, the default script file name is: start_dmgr.bat(sh) The script will be saved to the bin directory of the Deployment Manager installation.
`-timeout <seconds>`	Waiting time before server initialization times out and returns an error.
`-statusport <portnumber>`	Set the port number for server status callback.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of startServer.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-J-<java option>`	Options to be passed through to the Java interpreter. Options are specified in the form: -D<name>=<value>
`-help`	Prints a usage statement to the console.
`-?`	Prints a usage statement to the console. Same as the -help option.

A.1.3 Example

Example: A-1 startManager usage examples

```
$ cd /usr/WebSphere/DeploymentManager/bin

Start dmgr using defaults
```

```
$ startManager.sh

Start dmgr with tracing enabled

$ startManager.sh -trace

Start dmgr with output written to non-standard logfile

$ startManager.sh -logfile "../logs/dmgr/mylog.log"
```

A.2 stopManager

The **stopManager** command reads the configuration file for the Deployment Manager process. It sends a JMX command to the Deployment Manager telling it to shut down.

A.2.1 Notes

1. The command must be run from the bin directory of the Deployment Manager installation root: <WAS_ND_HOME>/bin. For example:

   /usr/WebSphere/DeploymentManager/bin

2. The command logs to the files shown in Table A-3.

Table A-3 stopManager logs

Log file	Content
<WAS_ND_HOME>/logs/dmgr/SystemOut.log	Console messages
<WAS_ND_HOME>/logs/dmgr/SystemErr.log	Console errors
<WAS_ND_HOME>/logs/dmgr/stopServer.log	Command output including trace and errors

3. By default, the command waits for the Deployment Manager to complete shutdown before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.

4. If the -host and -port arguments are not specified, then the command reads the Deployment Manager host name and IP port number from the serverindex.xml file located in the master repository:

   <WAS_ND_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/se rverindex.xml

5. If the -port argument is specified without -host, then the command assumes the host name is *localhost* and uses this host name with the port to communicate directly with the Deployment Manager. It does not need to read from the serverindex.xml file in the master repository.
6. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.2.2 Syntax

The syntax for the **stopManager** command is as follows:

```
stopManager.bat(sh) [options]
```

All arguments are optional. See Table A-4.

Table A-4 Options for stopManager

Option	Description
`-nowait`	Tells the command not to wait for successful shutdown of the Deployment Manager process.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to stopServer.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of stopServer.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-conntype <type>`	JMX connector to use for connection. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
`-host <hostname>`	The Deployment Manager JMX host name to use explicitly, so that configuration files do not have to be read to obtain the information.
`-port <portnumber>`	The Deployment Manager JMX port to use explicitly, so that configuration files do not have to be read to obtain the information.

Option	Description
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.2.3 Example

Example: A-2 stopManager usage examples

```
$ cd /usr/WebSphere/DeploymentManager/bin

Stop dmgr using defaults (SOAP, hostname and port read from master repository,
security disabled)

$ stopManager.sh

Stop dmgr when WebSphere security is enabled

$ stopManager.sh -username <user> -password <password>

Stop dmgr and have command return immediately

$ stopManager.sh -nowait

Stop dmgr using its RMI/IIOP JMX connector

$ stopManager.sh -conntype RMI -host <dmgr_host> -port <dmgr_port>
```

A.3 startNode

The **`startNode`** command reads the configuration file for the node agent process, and constructs a launch command to run it. Depending on the options that are specified, the command creates a new Java virtual machine (JVM) to run the agent process, or writes the launch command data to a file.

A.3.1 Notes

1. The command must be run from the bin directory of the node installation root: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-5.

Table A-5 startNode logs

Log file	Content
<WAS_HOME>/logs/nodeagent/SystemOut.log	Console messages
<WAS_HOME>/logs/nodeagent/SystemErr.log	Console errors
<WAS_HOME>/logs/nodeagent/startServer.log	Command output including trace and errors

3. The node agent process ID is written to:

   <WAS_HOME>/logs/nodeagent/nodeagent.pid

4. The node agent process on startup reads the serverindex.xml file of the Deployment Manager's node. The host name and IP port number read from this file are used to discover the Deployment Manager process:

   <WAS_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/serveri ndex.xml

5. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.3.2 Syntax

The syntax for the **`startNode`** command is as follows:

```
startNode.bat(sh) [options]
```

All arguments are optional. See Table A-6.

Table A-6 Options for startNode

Option	Description
`-nowait`	Tells the command not to wait for successful initialization of the launched node agent process.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.

Option	Description
-trace	Generates trace information into a file for debugging purposes. Output is to startServer.log.
-script [<script filename>]	Generate a launch script instead of starting the server. The script file name is optional. If the file name is not provided, the default script file name is: start_nodeagent.bat(sh) The script needs to be saved to the bin directory of the node installation.
-logfile <log file path>	Specifies alternative location for command's log output, instead of startServer.log. The path can be specified in the following forms: absolute, relative, or file name.
-replacelog	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
-username <username>	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
-password <password>	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
-timeout <seconds>	Waiting time before server initialization times out and returns an error.
-statusport <portnumber>	Set the port number for server status callback
-J-<java option>	Options to be passed through to the Java interpreter. Options are specified in the form: -D<name>=<value>.
-help	Prints a usage statement.
-?	Prints a usage statement. Same as the -help option.

A.3.3 Example

Example: A-3 startNode usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Start node agent using defaults (assumes security is disabled)

$ startNode.sh
```

```
Start node agent with tracing enabled

$ startNode.sh -trace

Start node agent when WebSphere security enabled

$ startNode.sh -username <user> -password <password>
```

A.4 stopNode

The **stopNode** command reads the configuration file for the node agent process and sends a JMX command to the node agent telling it to shut down.

A.4.1 Notes

1. The command must be run from the bin directory of the node installation root: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-7.

Table A-7 stopNode logs

Log file	Content
<WAS_HOME>/logs/nodeagent/SystemOut.log	Console messages
<WAS_HOME>/logs/nodeagent/SystemErr.log	Console errors
<WAS_HOME>/logs/nodeagent/stopServer.log	Command output including trace and errors

3. By default, the command waits for the node agent to complete shutdown before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.
4. If the -host and -port arguments are not specified, then the command reads the node agent host name and IP port number from the serverindex.xml file located in the node's configuration files:

   <WAS_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/serveri ndex.xml

5. If the -port argument is specified without -host, then the command assumes the host name is *localhost* and uses this host name with the port to

communicate directly with the node agent. It does not need to read from the serverindex.xml file.

6. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.4.2 Syntax

The syntax of the **stopNode** command is as follows:

```
stopNode.bat(sh) [options]
```

All arguments are optional. See Table A-8.

Table A-8 Options for stopNode

Option	Description
`-nowait`	Tells the command not to wait for successful shutdown of the node agent process.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to stopServer.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of stopServer.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-conntype <type>`	JMX connector to use for connection. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
`-host <hostname>`	The node agent JMX host name to use explicitly, so that configuration files do not have to be read to obtain the information.
`-port <portnumber>`	The node agent JMX port to use explicitly, so that configuration files do not have to be read to obtain the information.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.

Option	Description
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.4.3 Example

Example: A-4 stopNode usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Stop node agent using defaults (SOAP, hostname and port read from local
configuration file, WebSphere security disabled)

$ stopNode.sh

Stop node agent when WebSphere security is enabled

$ stopNode.sh -username <user> -password <password>

Stop node agent using its JMX RMI connector

$ stopNode.sh -conntype RMI -host <nodeagent host> -port <nodeagent port>
```

A.5 addNode

The **addNode** command adds a new node to an existing WebSphere Application Server V5 administrative cell (domain).

The actions performed by this command are:

1. Connects to Deployment Manager process - necessary for the file transfers performed to and from the Deployment Manager in order to add the node to the cell.
2. Attempts to stop all running application servers on the stand-alone node.
3. Backs up the current stand-alone node configuration to:

   <WAS_HOME>/config/backup/base/
4. Copies stand-alone node configuration to a new cell structure that matches the Deployment Manager structure at the cell level.

5. Creates a new local config directory and definition (server.xml) for the node agent.
6. Creates a separate jmsserver process containing the embedded messaging functionality. This is performed whether or not the base IBM WebSphere Application Server had embedded messaging installed.
7. Creates entries (directories and files) in the master repository for the new node's managed servers (node agent, jmsserver and application server(s)).
8. Uses the FileTransfer service to copy files from the new node to the master repository.
9. Uploads applications to the cell (only if the -includeapps option is specified).
10. Performs the first file synchronization for the new node. This pulls everything down from the cell to the new node.
11. Fixes the node's setupCmdLine and wsadmin scripts to reflect the new cell environment settings.
12. Launches the node agent.
13. Creates a queue manager for the new node.

Important: The following points must be borne in mind when adding a node to a cell.

- The cell must already exist.
- The cell's Deployment Manager must be running before addNode can be executed.
- The new node must have a unique name. If an existing node in the cell already has the same name, addNode will fail.
- By default, addNode does not carry over the applications (installed on the stand-alone node) when added to the cell. The -includeApps option must be used for this purpose.

A.5.1 Notes

1. The command must be run from the bin directory of the node installation root: <WAS_HOME>/bin. It cannot be run from the Deployment Manager. For example:

   /usr/WebSphere/AppServer/bin
2. The command logs to the files shown in Table A-9 on page 845.

Table A-9 addNode logs

Log file	Content
<WAS_HOME>/logs/addNode.log	Console messages and errors
<WAS_HOME>/logs/createMQ.<nodename>_jmsserver.log	Output resulting from creation of the node's WebSphere MQ queue manager and associated resources.

3. Depending on the size and location of the node being incorporated into the cell, the command can take a few minutes to complete.
4. The <dmgr_host> and <dmgr_port> arguments are mandatory. However, the port that needs to be specified depends upon the following situations:
   a. Which JMX connector to use

      If the -conntype option is not specified, then SOAP is assumed. In this instance the Deployment Manager's SOAP port (default is 8879) must be used. If the -conntype option is used to specify the RMI protocol, then it is the Deployment Manager's bootstrap port (default is 9809) that must be used.

   b. Whether non-default ports are in use

      If the Deployment Manager's SOAP and/or bootstrap port default values conflict with the ports used by other software, then the Deployment Manager software may have been configured to use other non-default ports.

> **Note:** The following file, located in the node's local configuration files, contains the IP ports in use by the Deployment Manager:
>
> <WAS_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/serverindex.xml

5. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.5.2 Syntax

The syntax of the **`addNode`** command is as follows:

```
addNode.bat(sh) <dmgr_host> <dmgr_port> [options]
```

The first two arguments are mandatory. For options, see Table A-10 on page 846.

Table A-10 Options for addNode

Option	Description
`-nowait`	Tells the command not to wait for successful completion of the node addition.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to addNode.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of addNode.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-conntype <type>`	JMX connector to use for connection. Valid values are SOAP or RMI. If not specified, SOAP is assumed. If RMI is specified, then the Deployment Manager's correct RMI/IIOP JMX connector port must be specified by the <dmgr_port> argument.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-includeapps`	Attempt to include the applications in the incorporation of the base node into a cell. Default is not to include the applications.
`-startingport <port>`	Used as the starting/base IP port number for the node agent and jmsserver processes created for this new node.
`-noagent`	Indicates that the new node agent (generated as part of adding the node to a cell) is not to be started at the end. Default is to start the node agent.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.5.3 Example

Example: A-5 addNode usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Add node to cell whose Deployment Manager has JMX SOAP connector on host
testhost and port 8879.

$ addNode.sh testhost 8879

Add node to cell whose Deployment Manager has JMX RMI connector on host host25
and port 9810. Also migrate all applications installed on the standalone node
to the cell.

$ addNode.sh host25 9810 -conntype RMI -includeApps

Add node to cell whose Deployment Manager has JMX SOAP connector on host hostx
and port 8879, when WebSphere security is enabled for the cell.

$ addNode.sh hostx 8879 -username <user> -password <password>
```

A.6 removeNode

`removeNode` detaches a node from a WebSphere Application Server V5 administrative cell (domain) and returns it to a base IBM WebSphere Application Server configuration.

The command performs the following operations:

1. Connects to the Deployment Manager process to read the configuration data.
2. Stops all of the running server processes of the node, including the node agent process.
3. Restores the backed up stand-alone node configuration - backed up when the node was originally added to the cell.

   Note: This loses all configuration changes made to the managed servers since the node was originally added to the cell.

4. Removes the node's configuration from the master repository of the cell. The local copy of the repository held on each node will get updated at the next synchronization point for each node agent. Although the complete set of configuration files are not pushed out to other nodes, some directories and files are pushed out to all nodes.

5. Removes installed applications from application servers in the cell that are part of the node being removed.
6. Deletes the node's queue manager.
7. Copies the original application server cell configuration into the active configuration.

A.6.1 Notes

1. The command must be run from the bin directory of the node installation root: <WAS_HOME>/bin. It cannot be run from the Deployment Manager. For example:

   /usr/WebSphere/AppServer/bin
2. The command logs to the files shown in Table A-11.

Table A-11 removeNode logs

Log file	Content
<WAS_HOME>/logs/removeNode.log	Console messages and errors
<WAS_HOME>/logs/deleteMQ.<nodename>_jmsserver.log	Output resulting from deletion of the node's WebSphere MQ queue manager and associated resources.

3. Depending upon the size and location of the node, the command can take a few minutes to complete.
4. To decouple an application server from distributed administration, while retaining its configuration (for example, applications), use the admin client to set the server's "standalone" property to true.
5. The command reads the Deployment Manager host name and IP port number from the serverindex.xml file located in the node's local configuration files:

   <WAS_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/serverindex.xml
6. Unlike the **`addNode`** command, **`removeNode`** always uses the SOAP JMX connector of the Deployment Manager. There is no option provided for specifying the RMI JMX connector.
7. The command provides the -force option to force the local node's configuration to be decoupled from the cell even if the Deployment Manager cannot be contacted. However, if this situation occurs the cell's master repository will then have to be separately updated to reflect the node's

removal, for example through manual editing of the master repository configuration files.

8. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.6.2 Syntax

The syntax of the **removeNode** command is as follows:

```
removeNode.bat(sh) [options]
```

All the arguments are optional. See Table A-12.

Table A-12 Options for removeNode

Option	Description
`-nowait`	Tells the command not to wait for successful removal of the node.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to removeNode.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of removeNode.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-force`	Cleans up the local node configuration regardless of whether the Deployment Manager can be reached for cell repository cleanup.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.6.3 Example

Example: A-6 removeNode usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Remove the node (WebSphere security disabled)

$ removeNode.sh

Remove the node (WebSphere security enabled)

$ removeNode.sh -username <user> -password <password>

Remove the node even if the Deployment Manager cannot be contacted

$ removeNode.sh -force
```

A.7 syncNode

The **`syncNode`** command is used to force the synchronization of a node's local configuration repository with the master repository on the Deployment Manager node.

A.7.1 Notes

1. The command must be run from the bin directory of the node installation root: <WAS_HOME>/bin. It cannot be run from the Deployment Manager. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-13.

Table A-13 syncNode logs

Log file	**Content**
<WAS_HOME>/logs/syncNode.log	Console messages and errors

3. By default, the command waits for the completion of the node synchronization before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.
4. The command communicates directly with the Deployment Manager, bypassing the local node agent process. This is necessary so that the node

agent's configuration can be updated as necessary during the synchronization.

5. The command makes a synchronization request through the Deployment Manager's JMX connector specified on the command line.
6. If the -conntype option is not specified, then the SOAP JMX connector is assumed.
7. If the <dmgr_port> option is specified, then it must specify the correct port for the chosen JMX connector. If <dmgr_port> is not specified, then the local node's serverindex.xml configuration file is read to retrieve the port for the Deployment Manager's chosen JMX connector:

   <WAS_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/serveri ndex.xml
8. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.7.2 Syntax

The syntax of the **syncNode** command is as follows:

```
syncNode.bat(sh) <dmgr_host> [dmgr_port] [options]
```

The first argument is mandatory. The options are listed in Table A-14.

Table A-14 Options for syncNode

Option	Description
`-nowait`	Tells the command not to wait for successful synchronization of the node.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to syncNode.log.
`-conntype <type>`	JMX connector type to use for connection to the Deployment Manager. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
`-stopservers`	Indicates that the node agent and all managed servers of the node should be stopped prior to synchronizing the node's configuration with the cell.

Option	Description
`-restart`	Indicates that the node agent is to be restarted after synchronizing the node's configuration with the cell.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of syncNode.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.7.3 Example

Example: A-7 syncNode usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Synchronize node configuration with cell whose Deployment Manager's JMX SOAP
connector is on host host25, port 8879 (WebSphere security disabled)

$ syncNode.sh host25 8879

Synchronize node configuration with cell whose Deployment Manager's JMX SOAP
connector is on host host25, port 8879 (WebSphere security enabled)

$ syncNode.sh host25 8879 -user <username> -password <password>

Synchronize node configuration with cell whose Deployment Manager's JMX RMI
connector is on host hostx, port 9000 (WebSphere security enabled)

$ syncNode.sh hostx 9000 -conntype RMI -user <username> -password <password>

Synchronize node configuration with cell whose Deployment Manager's JMX RMI
connector is on host hostx, port 9000 (WebSphere security enabled). Ensure
tall servers on the node are stopped prior to synchronization and that the node
agent is restarted on completion.
```

```
$ syncNode.sh hostx 9000 -conntype RMI -user <username> -password <password>
-stopservers -restart
```

A.8 cleanupNode

cleanupNode cleans up a node configuration from a partially created network deployment distributed administration cell.

Important: Use this command only if a partial **addNode** command or **removeNode** command fails to remove all configuration data from the administration configuration

A.8.1 Notes

1. The command must be run from the bin directory of the Deployment Manager: <WAS_ND_HOME>/bin. It cannot be run from the node to be cleaned up. For example:

   /usr/WebSphere/DeploymentManager/bin
2. The command logs to the files shown in Table A-15.

Table A-15 cleanupNode logs

Log file	Content
<WAS_ND_HOME>/logs/cleanupNode.log	Console messages and errors

3. By default, the command waits for the completion of node cleanup before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.
4. The command does not provide a -conntype option. SOAP is assumed.
5. If the <dmgr_host> and/or <dmgr_port> arguments are not specified, then the command reads the Deployment Manager's JMX SOAP connector details from the serverindex.xml in the master repository:

   <WAS_ND_HOME>/config/cells/<cellname>/nodes/<dmgr_host>Manager/serverindex.xml
6. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.8.2 Syntax

The syntax of the **`cleanupNode`** command is as follows:

```
cleanupNode.bat(sh) <node name> <dmgr_host> <dmgr_port> [options]
```

The first argument is mandatory. The options are listed in Table A-16.

Table A-16 Options for cleanupNode

Option	Description
`-nowait`	Tells the command not to wait for successful cleanup of the node.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to cleanupNode.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of cleanupNode.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.8.3 Example

Example: A-8 cleanupNode usage examples

```
$ cd /usr/WebSphere/DeploymentManager/bin

Cleanup node nodeXYZ (WebSphere security disabled)

$ cleanupNode.sh nodeXYZ

Cleanup node nodeXYZ (WebSphere security enabled)
```

```
$ cleanupNode.sh nodeXYZ -username <user> -password <password>
```

A.9 startServer

The **startServer** command reads the configuration file for the specified server process and constructs a launch command for the server. Depending on the options you specify, a new Java virtual machine (JVM) can be launched to run the server process or to write the launch command data to a file.

A.9.1 Notes

1. The command must be run from the bin directory of the server installation root: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-17.

Table A-17 startServer logs

Log file	Content
<WAS_HOME>/logs/<servername>/SystemOut.log	Console messages
<WAS_HOME>/logs/<servername>/SystemErr.log	Console errors
<WAS_HOME>/logs/<servername>/startServer.log	Command output including trace and errors

3. By default, the command waits for the server to complete startup before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.
4. The command reads the node agent's discovery port number from the serverindex.xml file located in the node's local configuration files:

   <WAS_HOME>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml

5. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.9.2 Syntax

The syntax of the **StartServer** command is as follows:

```
startServer.bat(sh) <server> [options]
```

where `<server>` is the name of the server to be started. The first argument is mandatory. The options are listed in Table A-18.

Table A-18 Options for startServer

Option	Description
`-nowait`	Tells the command not to wait for successful startup of the server.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to startServer.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of startServer.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-script [<script filename>]`	Generate a launch script instead of starting the server. The script file name is optional. If the file name is not provided, the default script file name is start_<server>. The script needs to be saved to the bin directory of the node installation.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-timeout <seconds>`	Waiting time before server initialization times out and returns an error.
`-statusport <portnumber>`	Set the port number for server status callback
`-J-<java option>`	Options to be passed through to the Java interpreter. Options are specified in the form: -D<name>=<value>.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.9.3 Example

Example: A-9 startServer usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Start the application server server1 (WebSphere security disabled)

$ startServer.sh server1

Start the application server server1 (WebSphere security enabled)

$ startServer.sh server1 -username <user> -password <password>

Start the jmsserver

$ startServer.sh jmsserver
```

A.10 stopServer

The **stopServer** command reads the configuration file for the specified server process. It sends a JMX command to the server telling it to shut down.

A.10.1 Notes

1. The command must be run from the bin directory of the server installation root: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-19 on page 857.

Table A-19 stopServer logs

Log file	Content
<WAS_HOME>/logs/<servername>/SystemOut.log	Console messages
<WAS_HOME>/logs/<servername>/SystemErr.log	Console errors
<WAS_HOME>/logs/<servername>/stopServer.log	Command output including trace and errors

3. By default, the command waits for the server to complete shutdown before it returns control to the command line. There is a -nowait option to return immediately, as well as other options to control the behavior of the command.

4. The command sends a shutdown request to the server using either the SOAP or RMI JMX connector of the server. The default (if -conntype option is not specified) is SOAP.
5. The **stopServer** command is only for stopping a server local to the node installation in which the command is executed. As such, the command does not provide a -host option, since a host name of *localhost* is assumed.
6. If the -port option is not specified, then the command reads the server's appropriate JMX connector port number from the serverindex.xml file on the node:

   <WAS_HOME>/config/cells/<cellname>/nodes/<nodename>/serverindex.xml
7. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.10.2 Syntax

The syntax of the **stopServerNode** command is as follows:

```
stopServer.bat(sh) <server> [options]
```

where `<server>` is the name of the server to be stopped. The first argument is mandatory. The options are listed in Table A-20.

Table A-20 Options for stopServer

Option	Description
`-nowait`	Tells the command not to wait for successful stop of the server.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to stopServer.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of stopServer.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.

Option	Description
`-timeout <seconds>`	Waiting time before server initialization times out and returns an error.
`-conntype <connector type>`	Type of JMX connector to use for connection to the Deployment Manager. Valid values are SOAP or RMI. If not specified, SOAP is assumed.
`-port <portnumber>`	The server JMX port to use explicitly, so that configuration files do not have to be read to obtain the information.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.10.3 Example

Example: A-10 stopServer usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Stop the application server server1 (WebSphere security disabled)

$ stopServer.sh server1

Stop the application server server1 (WebSphere security enabled)

$ stopServer.sh server1 -username <user> -password <password>

Stop server server2 using its JMX RMI connector on port 10000, with tracing
enabled.

$ stopServer.sh server2 -conntype RMI -port 10000 -trace
```

A.11 serverStatus

`serverStatus` obtains the status of one or all of the servers configured on a node.

A.11.1 Notes

1. The command must be run from the bin directory of the server installation root: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-21, depending upon whether the -all (all servers on node) or servername (single server) argument is used:

Table A-21 serverStatus logs

Log file	Content
<WAS_HOME>/logs/<servername>/serverStatus.log	If server name (not -all) argument is used.
<WAS_HOME>/logs/serverStatus.log	If -all argument is used.

3. The command reads the node's local serverindex.xml configuration file to determine the JMX connector ports for each of the servers on the node. The command uses this information to make a status request to one (or all) servers on the node via the each server's JMX SOAP connector.
4. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.11.2 Syntax

The syntax of the **`ServerStatus`** command is as follows:

```
serverStatus.bat(sh) <server>|-all [options]
```

The first argument is mandatory. The argument is either the name of the server for which status is desired, or the -all keyword, which requests status for all servers defined on the node.

Table A-22 Options for serverStatus

Option	Description
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of serverStatus.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.

Option	Description
`-trace`	Generates trace information into a file for debugging purposes. Output is to serverStatus.log.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.11.3 Example

Example: A-11 serverStatus usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Check the status of server server1 (WebSphere security disabled)
$ serverStatus.sh server1

Check the status of server server1 (WebSphere security enabled)
$ serverStatus.sh server1 -username <user> -password <password>

Check the status of all servers on the local node (including the node agent and
jmsserver)
$ serverStatus.sh -all
```

A.12 createmq

`createmq` creates the messaging broker, queue manager and supporting messaging objects for a node.

The actions performed by this command are:

1. Uses the WebSphere MQ **`crtmqm`** command to create on the local machine a new queue manager called the following:

   WAS_<cell name>_<node name>

2. Uses the WebSphere MQ **`strmqm`** command to start the new WebSphere MQ queue manager.

3. Uses the WebSphere MQ **runmqsc** command and the WebSphere MQ script located at:

   <WAS_HOME>/bin/createmq.mqsc

   to perform the following adjustments to the queue manager:

   a. Create new queues to support JMS point-to-point and publish/subscribe messaging.
   b. Create new channel definition.
   c. Delete all standard channels.
   d. Delete all default queues.

4. Uses the **wempsdeletebroker** command to delete any existing broker of the following name on the local machine:

   WAS_<cell name>_<node name>

5. Uses the **wempscreatebroker** command to create a new broker of the following name and associate it with the queue manager:

   WAS_<cell name>_<node name>

6. Registers the broker as a Windows service (on Windows only).
7. Uses the WebSphere MQ **endmqm** command to stop the new queue manager.

A.12.1 Notes

1. The command must be run from the bin directory of the server installation root: <WAS_HOME>/bin. It cannot be run from the Deployment Manager installation. For example:

   /usr/WebSphere/AppServer/bin

2. The command must be run after the installation of WebSphere MQ, the broker and the JMS client (MA88) classes. The command assumes the paths have been configured correctly.
3. The command logs to the files shown in Table A-23 on page 862.

Table A-23 createmq logs

Log file	Content
<WAS_HOME>/logs/createMQ_<nodename>_<servername>.log	For stand-alone base node installations
<WAS_HOME>/logs/createMQ_<nodename>_<jmsserver>.log	For nodes added to a Network Deployment cell.

4. The **createmq** command script is specific to each platform (for example AIX, Windows etc.), but createmq.mqsc is common to all platforms.

A.12.2 Syntax

The syntax of the **createmq** command is as follows:

```
createmq.bat(sh) <WAS_HOME> <Cell name> <Node name> <Server name> <mq_home>
<Broker root>
```

All arguments are mandatory. The options are listed in Table A-24.

Table A-24 Options for createmq

Option	Description
`WAS root`	The root WebSphere Application Server installation directory of the node, for example /usr/WebSphere/AppServer.
`Cell name`	Name of the cell, for example hostxNetwork.
`Node name`	Name of the local node.
`Server name`	Name of the managed server in which to install and configure embedded messaging. For Network Deployment configurations, the server on each node is always *jmsserver*.
`WebSphere MQ root`	Root directory of the local WebSphere MQ (previously MQSeries) installation, for example /usr/IBM/MQSeries.
`Broker root`	Root directory of the WebSphere MQ (previously MQSeries) publish/subscribe broker software, for example /usr/IBM/pubsub.

A.12.3 Example

Example: A-12 createmq usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Create MQ resources for a standalone server server1 of a WAS-base installation
with nodename of node1

$ createmq.sh "/usr/WebSphere/AppServer" Node1 Node1 server1
"/usr/IBM/MQSeries" "/usr/IBM/pubsub"
```

```
Create MQ resources for the jmsserver of a Network Deployment node. The cell is
cell1 and node is node2.

$ createmq.sh "/usr/WebSphere/AppServer" cell1 node2 jmsserver
"/usr/IBM/MQSeries" "/usr/IBM/pubsub"
```

A.13 deletemq

deletemq deletes the messaging broker, queue manager and supporting messaging objects for a node.

A.13.1 Notes

1. The command must be run from the bin directory of the server installation root: <WAS_HOME>/bin. It cannot be run from the Deployment Manager installation. For example:

   /usr/WebSphere/AppServer/bin

2. The command logs to the files shown in Table A-25.

Table A-25 deletemq logs

Log file	Content
<WAS_HOME>/logs/deleteMQ_<nodename>_<servername>.log	For stand-alone base node installations
<WAS_HOME>/logs/deleteMQ_<nodename>_<jmsserver>.log	For nodes added to a Network Deployment cell.

3. The command does not uninstall the WebSphere MQ or broker products.

A.13.2 Syntax

The syntax of the **deletemq** command is as follows:

```
deletemq.bat(sh) <Cell name> <Node name> <Server name>
```

All arguments are mandatory. The options are listed in Table A-26.

Table A-26 Options for deletemq

Option	Description
`Cell name`	Name of the cell.

Option	Description
`Node name`	Name of the local node.
`Server name`	Name of the managed server from which embedded messaging is to be removed. For Network Deployment configurations, the server on each node is always *jmsserver*.

A.13.3 Example

Example: A-13 deletemq usage examples

```
$ cd /usr/WebSphere/AppServer/bin

Delete the MQ resources for a standalone server server1 of a WAS-base
installation with nodename of node1

$ deletemq.sh node1 node1 server1

Delete the MQ resources for the jmsserver of a Network Deployment node. The
cell is cell1 and node is node2.

$ deletemq.sh cell1 node2 jmsserver
```

A.14 backupConfig

The **`backupConfig`** command backs up the node configuration to a ZIP archive file.

The actions performed by this command are:

1. Stops all running processes (node agent, application servers and JMS Server) of the local node.
2. Archives the contents of the <WAS_HOME>/config directory (and subdirectories) to a named ZIP file, or WebSphereConfig_<yyyy-mm-dd>.zip, created in the same directory in which the command is run.

A.14.1 Notes

1. The command must be run from the bin directory of the IBM WebSphere Application Server installation root of the node to be backed up: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

2. The command does not support backup of remote nodes from a central location, for example the Deployment Manager machine. The master repository can be backed up by running **`backupConfig`** on the node hosting the Deployment Manager, for example

   /usr/WebSphere/DeploymentManager/bin/backupConfig.sh ...

3. The command logs to the files shown in Table A-27.

Table A-27 backupConfig logs

Log file	**Content**
<WAS_HOME>/logs/backupConfig.log	Command output and errors.

4. If specified, the backup archive file name does not require the zip file extension. The command will create an archive named <filename>.zip.
5. The command does not restart the processes that were stopped (if any) to perform the backup. These processes will require a manual restart by the administrator.
6. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.14.2 Syntax

The syntax of the **`backupConfig`** command is as follows:

```
backupConfig.bat(sh) <backup_file> [options]
```

where `<backup_file>` specifies the name of the ZIP file to which the backup is written. If not specified, a unique file name is automatically generated and used. All arguments are optional. The options are listed in Table A-28 on page 867.

Table A-28 Options for backupConfig

Option	Description
`-nostop`	Do not stop the servers before backing up the configuration.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to backupConfig.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of backupConfig.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.14.3 Example

Example: A-14 backupConfig usage examples

```
Backup the configuration of a node (WebSphere security disabled) to file
mybackup

$ cd /usr/WebSphere/AppServer/bin
$ backupConfig.sh mybackup

Backup the configuration of a node (WebSphere security enabled) to file zzz

$ cd /usr/WebSphere/AppServer/bin
$ backupConfig.sh zzz -username <user> -password <password>

Backup the cell's master repository to default ZIP archive, without stopping
any processes (WebSphere security enabled)
```

```
$ cd /usr/WebSphere/DeploymentManager/bin
$ backupConfig.sh -nostop -username <user> -password <password>
```

A.15 restoreConfig

The **restoreConfig** command restores the configuration of the node after backing up the configuration using the **backupConfig** command.

The actions performed by this command are:

1. Stops all running processes (node agent, application servers and JMS Server) of the local node. This prevents a node synchronization from occurring during the install.
2. If the <WAS_HOME>/config configuration directory already exists, it is renamed to config.old.
3. Restores the contents of the archive to <WAS_HOME>/config on the local node.

A.15.1 Notes

1. The command must be run from the bin directory of the IBM WebSphere Application Server installation root of the node to be restored: <WAS_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin
2. The command does not support restoring remote nodes from a central location, for example the Deployment Manager machine. The master repository can be restored up by running **restoreConfig** on the node hosting the Deployment Manager, for example

   /usr/WebSphere/DeploymentManager/bin/restoreConfig.sh ...
3. The command logs to the files shown in Table A-29.

Table A-29 restoreConfig logs

Log file	Content
<WAS_HOME>/logs/restoreConfig.log	Command output and errors.

4. The command does not restart the processes that were stopped (if any) to perform the restore. These processes will require a manual restart by the administrator.

5. If WebSphere security is enabled, the -username and -password arguments are mandatory. The specified user name and password must be present in the user directory used by the WebSphere security subsystem.

A.15.2 Syntax

The syntax of the **restoreConfig** command is as follows:

```
restoreConfig.bat(sh) <archive_file> [options]
```

where `<archive_file>` specifies the backup archive to be restored. The first argument is mandatory. The options are listed in Table A-30.

Table A-30 Options for restoreConfig

Option	Description
`-nostop`	Do not stop the servers before restoring the configuration.
`-quiet`	Suppresses progress information printed to console in normal mode. This option does not affect information written to file.
`-trace`	Generates trace information into a file for debugging purposes. Output is to restoreConfig.log.
`-logfile <log file path>`	Specifies alternative location for command's log output, instead of restoreConfig.log. The path can be specified in the following forms: absolute, relative, or file name.
`-replacelog`	Starts a new log, replacing any previous log of the same name. If this argument is not specified, the default behavior is to append output to the existing file.
`-username <username>`	User name for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-password <password>`	Password for authentication if WebSphere security is enabled. Ignored if WebSphere security is disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.15.3 Example

Example: A-15 restoreConfig usage examples

```
Restore the configuration of a node (WebSphere security disabled) from file
mybackup

$ cd /usr/WebSphere/AppServer/bin
$ restoreConfig.sh mybackup

Restore the configuration of a node (WebSphere security enabled) from file zzz

$ cd /usr/WebSphere/AppServer/bin
$ restoreConfig.sh zzz -username <user> -password <password>

Restore the cell's master repository from file yyy, without stopping any
processes (WebSphere security enabled)

$ cd /usr/WebSphere/DeploymentManager/bin
$ restoreConfig.sh yyy -nostop -username <user> -password <password>
```

A.16 wsinstance

The **wsinstance** command creates a new and independent configuration instance of a IBM WebSphere Application Server base installation on one machine.

The actions performed by this command are:

1. Loads the instance.xml build file from the same directory in which the command is run. This file contains a definition of all the components to be copied or generated for a new node instance.
2. The following directories are generated for the new instance and files are copied from the original instance. The only files required are those that distinguish the new instance from the original.
   - bin

     Contains a new setupCmdLine.bat(sh) to configure the new instance's environment.
   - config

     Complete copy of the original instance's config directory (and subdirectories), but with new node and server1 (default server).
   - etc

     Complete copy of the original instance's etc directory.

- installableApps
- installedApps
- logs
- properties

  Complete copy of the original instance's properties directory.
- temp
- translog
- wstemp

3. Creates the <hostname>_<instancename>_portdef.props file in the same directory where the command is run. This file contains the list of port numbers assigned for the new instance. For example:

```
BOOTSTRAP_ADDRESS=2810
SOAP_CONNECTOR_ADDRESS=8881
DRS_CLIENT_ADDRESS=7874
INTERNAL_JMS_SERVER=5560
JMSSERVER_QUEUED_ADDRESS=5561
JMSSERVER_DIRECT_ADDRESS=5562
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=0
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=0
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=0
HTTP_TRANSPORT=9081
HTTP_TRANSPORT_ADMIN=9091
HTTPS_TRANSPORT=9444
HTTPS_TRANSPORT_ADMIN=9044
```

4. Creates a new queue manager and supporting objects for the embedded messaging of the new instance. Uses **`createmq`** and logs all output to createMQ_<hostname>_<instancename>.server1.log.

A.16.1 Notes

1. The command must be run from the wsinstance directory of the IBM WebSphere Application Server installation: <WAS_HOME>/bin/wsinstance. It cannot be run from the Deployment Manager installation. For example:

   /usr/WebSphere/AppServer/bin/wsinstance/wsinstance.sh ...
2. The command logs to the files shown in Table A-31 on page 872.

Table A-31 wsinstance logs

Log file	Content
<WAS_HOME>/bin/wsinstance/installAdmin_<hostname>_<instancename>.log	Output generated during installation of adminconsole application into the default (server1) server of the new node instance.
<WAS_HOME>/bin/wsinstance/createMQ_<hostname>_<instancename>.server1.log	Output generated during execution of **`createmq`** to create MQ resources for the default (server1) server of the new node instance

3. The new instance has its own root directory containing its own repository (under the config subdirectory) as well as the minimum set of directories and files to support application servers and applications.
4. All instances share the following components with the initial instance:
   - Runtime scripts
   - Libraries
   - Java Development Kit (JRE)
5. The bin directory of each new instance has a node-specific version of the setupCmdLine script. In order to use any of the command-line tools with the new node, the user must:
   a. Change directory to the new node instance's bin directory.
   b. Invoke the required command, for example **`startServer`**. This invokes the command from the original installation's bin directory, but uses the new instance's setupCmdLine script to configure the environment correctly for the new instance.
6. Applications installed on the base installation are not carried over into the new instance. This includes the WebSphere administrative console application.
7. **`addNode`** must still be used to add the new node instance to an existing Network Deployment cell.
8. All instances can be configured and executed independently.
9. The **`uninstall`** command cannot be used to remove a configuration instance. Instead configuration instances can only be deleted using the -delete option of the **`wsinstance`** command.
10. To uninstall all instances on a server, the **`wsinstance`** command must be used to delete each configuration instance in turn until only the original installation

instance remains. At this point, the **uninstall** command can be used to uninstall the software.

A.16.2 Syntax

The syntax of the **wsinstance** command is as follows:

```
wsinstance.bat(sh) -name instanceName -path instancePath -host hostName
[-startingPort startingPort] -create|-delete  [-debug]
```

The options are listed in Table A-32.

Table A-32 Options for wsinstance

Option	Description
`-name <instancename>`	Unique name for this instance on this host.
`-path <instancepath>`	Path under which the new instance is to be created. Takes the places of <WAS_HOME>. The specified directory will contains a bin, config, properties etc directories.
`-host <hostname>`	Host name of the machine. Used to encode a unique node name for the instance, as <hostname>_<instancename>
`-startingPort <port>`	Port number to use as a basis for the port numbers assigned to the new instance.
`-create | delete`	Determines whether to create a new base instance or delete an existing instance.
`-debug`	Generates debugging information.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.16.3 Example

Example: A-16 wsinstance usage examples

```
$ cd /usr/WebSphere/AppServer/bin/wsinstance

Create a new node instance inst2

$ wsinstance.sh -name inst2 -path “/usr/apps/myinst1” -host myhost -create

Create a new node instance inst3 where IP ports are allocated starting at 20000
```

```
$ wsinstance.sh -name inst2 -path "/usr/apps/myinst1" -host myhost
-startingpoint 20000 -create

Delete the node instance inst2

$ wsinstance.sh -name inst2 -host myhost -delete
```

A.17 GenPluginCfg

The **GenPluginCfg** command reads the contents of the configuration repository on the local node to generate the Web server plug-in configuration file.

A.17.1 Notes

1. The command must be run from the bin directory of either the base or Deployment Manager installation root: <WAS_HOME>/bin or <WAS_ND_HOME>/bin. For example:

   /usr/WebSphere/AppServer/bin

   or

   /usr/WebSphere/DeploymentManager/bin
2. The command accesses the contents of the local configuration repository (<WAS_HOME>/config) directly from the file system. It does not access the repository indirectly through a Deployment Manager, node agent or application server process. As a consequence the command can be invoked without any WebSphere V5.0 processes running on the machine.
3. Only those processes that meet the following criteria can be included in the generated plug-in configuration:
   - Servers with a server definition file (server.xml) in the local configuration repository.

     If the command is invoked from the bin directory of a node installation (rather than from the Deployment Manager bin directory), then only those application servers that are part of that node will have a server.xml under the node's configuration repository.
   - Servers with configured Web containers and HTTP transports, that is Deployment Manager and application servers.
4. The command provides options to generate plug-in configuration files with contents restricted to different levels (scope) of the configuration repository and below:

   – Cell level

     Contains one <ServerCluster> XML stanza for each of the following: the Deployment Manager server (dmgr), each cluster configured in the cell, and each stand-alone (non-clustered) application server for all nodes within the cell.

     All application servers that are members of a cluster will be listed as <Server> XML stanzas under the single <ServerCluster> XML stanza representing the cluster.

     Node agents are not included in the generated output since they do not have a Web container (and HTTP transports) configured, and therefore cannot be sent requests by the Web server plug-in.

   – Node level

     Contains one <ServerCluster> XML stanza for each application server within the node.

     Application servers that are cluster members are not distinguished from non-clustered application servers. Therefore unlike the cell-level plug-in configuration, a node-level configuration will have a separate <ServerCluster> XML stanza for each server.

     Node agents are not included in the generated output since they do not have a Web container (and HTTP transports) configured, and therefore cannot be sent requests by the Web server plug-in.

   – Server level

     Contains just the one <ServerCluster> XML stanza for the application server specified.

5. The node and server-level options provide the ability to generate plug-in configurations that do not contain <ServerCluster> stanzas for all application servers in a cell. This allows the contents of the configuration file to be tailored to include only those nodes or specific application servers to be served requests by a Web server, without having to manually edit the file to remove unwanted server entries. For example, the node-level option can be used to generate the configuration necessary for a local Web server to serve requests only to those servers on the local node.
6. Although the administrative console can be used to generate the Web server plug-in configuration file, it only generates a cell-level plug-in configuration.
7. The command logs all messages and errors to the console.

A.17.2 Syntax

The syntax of the **GenPluginCfg** command is as follows:

```
:GenPluginCfg.bat(sh) [options]
```

All options are optional.The options are listed in Table A-33.

Table A-33 Options for GenPluginCfg

Option	Description
`-config.root <config root>`	Specifies the directory path of the particular configuration repository to be scanned. Defaults to the value of CONFIG_ROOT defined in the SetupCmdLine.bat(sh) script.
`-cell.name <cell name>`	Restricts generation to only the named cell in the configuration repository. Defaults to the value of WAS_CELL defined in the SetupCmdLine.bat(sh) script.
`-node.name <node name>`	Restricts generation to only the named node in the particular cell of the configuration repository. Defaults to the value of WAS_NODE defined in the SetupCmdLine.bat(sh) script.
`-server.name <server name>`	Restricts generation to only the named server in the particular node and cell of the configuration repository. There is no default. If this option is used, then the plug-in configuration generated will only contain a ServerCluster XML stanza for this server.
`-output.file.name <filename>`	Defines the path to the generated plug-in configuration file. Defaults to <WAS_ROOT>/config/cells/plugin-cfg.xml
`-debug <yes | no>`	Enables or disables output of debugging messages. Default is no, ie. debug disabled.
`-help`	Prints a usage statement.
`-?`	Prints a usage statement. Same as the -help option.

A.17.3 Examples

Example A-17 shows how to generate a a cell-level plug-in configuration, stored in the default location: <WAS_ROOT>/config/cells/plugin-cfg.xml.

Example: A-17 Generate cell -evel plug-in configuration

```
$ cd /usr/WebSphere/AppServer/bin
$ GenPluginCfg.sh
```

Example A-18 shows how to generate a cell-level plug-in configuration, saved to a non-default location: <WAS_ROOT>/config/cells/plugin-cfg2.xml.

Example: A-18 Generate cell-level plug-in configuration in non-default location

```
$ cd /usr/WebSphere/AppServer/bin
$ GenPluginCfg.sh -cell.name <cellname> -output.file.name plugin-cfg2.xml
```

Example A-19 shows how to generate a node-level plug-in configuration for nodeA, stored in a non-default location:<WAS_ROOT>/temp/plugin-cfg.xml.

Example: A-19 Generate node-level plug-in configuration

```
$ cd /usr/WebSphere/AppServer/bin
$ GenPluginCfg.sh -cell.name <cellname> -node.name nodeA -output.file.name
<WAS_ROOT>/temp/plugin-cfg.xml
```

Example A-20 shows how to generate a server-level plug-in configuration for server1 on nodeA, stored in the default location: <WAS_ROOT>/config/cells/plugin-cfg.xml.

Example: A-20 Generate server-level plug-in configuration

```
$ cd /usr/WebSphere/AppServer/bin
$ GenPluginCfg.sh -cell.name <cellname> -node.name nodeA -server.name server1
```

B

Application Server Toolkit

The Application Server Toolkit (referred to in this text as the “toolkit”) provides the basics for dealing with applications once the development process is over, including packaging for deployment, debugging running applications, profiling for performance, and analyzing runtime component logs.

The Application Servet Toolkit is based on the Eclipse Workbench and thus has the same look and feel as the WebSphere Studio tools. The toolkit consists of a subset of the same features you will find in WebSphere Studio products.

Using the Eclipse-based Workbench tool provides many usability features:

- An intuitive GUI interface with the same look and feel and many of the same functions as WebSphere Studio.
- Persistent settings that allow you to customize the Workbench to suit your needs and to start where you left off when you open the toolkit.
- Search capabilities for files or text within files.
- Automatic validation features.
- The ability to create your own tools as plug-ins.
- Progress indicator, drag and drop features, preference settings, customizable perspectives, and many other usability features.

This appendix will give you a quick introduction to the toolkit and how it can be used.

Installation and platform support

The toolkit is shipped with WebSphere Application Server V5. It is packaged on its own CD-ROM and installed separately from the application server. The toolkit contains the following components:

- Toolkit Workbench Base
- Assembly Toolkit Component
- Debug Component, including the Log Analyzer and profiling tools

These can be installed separately, but dependencies exist. Both the Assembly Toolkit and the Debug Component require the Workbench base. The Assembly Toolkit requires the Debug Component.

After the toolkit installs, you will be given the opportunity to install the IBM Agent Controller. The Agent Controller is a daemon process that runs on an application server host. It enables the toolkit to interact with the application servers remotely. If you will be using the profiling or debugging components, you will need to install the Agent Controller on each application server host with which you will be working.

The Application Server Toolkit is supported on the following platforms:

- Win32
- Linux on Intel
- HP-UX
- AIX
- Solaris

Note: The Assembly Toolkit component is only available on Windows and Linux. The Debug Component, Toolkit Workbench Base, and the IBM Agent Controller are available on all platforms listed.

Using the toolkit

Note: Since the functionality is limited on platforms other than Windows and Linux, the rest of this chapter will assume that you are using a Windows system. If you are installing on one of the other platforms, some of these features may not be available.

To start the toolkit:

1. Go to the <astk_install> directory. During the install process, you will have the opportunity to take the default or define the install directory. On Windows, the default install directory is c:\Program Files\IBM\ASTK.
2. Enter the following:

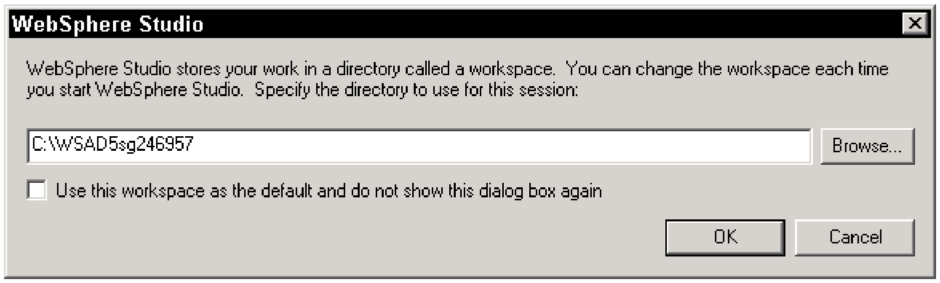
   ```
   astk
   ```

Specifying a workspace

The workspace is a private work area used to hold the project files and metadata for use by the toolkit. The first time you start the toolkit, you will be prompted for the location of the workspace to use. The default workspace location is:

```
<My Documents>\IBM\astk\workspace
```

You can use this workspace or enter a directory to use. If the directory does not exist, it will be created for you.


WebSphere Studio

WebSphere Studio stores your work in a directory called a workspace. You can change the workspace each time you start WebSphere Studio. Specify the directory to use for this session:

C:\WSAD5sg246957

☐ Use this workspace as the default and do not show this dialog box again

Figure B-1 Selecting a workspace

If you select the check box **Use this workspace as the default...**, you will not be prompted again. The workspace location you enter will be used whenever you start the toolkit. If later, you want the prompt to reappear, start the toolkit with the `-setworkspace` option:

```
cd <astk_install>
astk -setworkspace
```

You can change the default workspace without using the prompt by starting the toolkit with the `-data` parameter and entering the workspace directory to use.

```
cd <astk_install>
astk -data c:\myworkspace
```

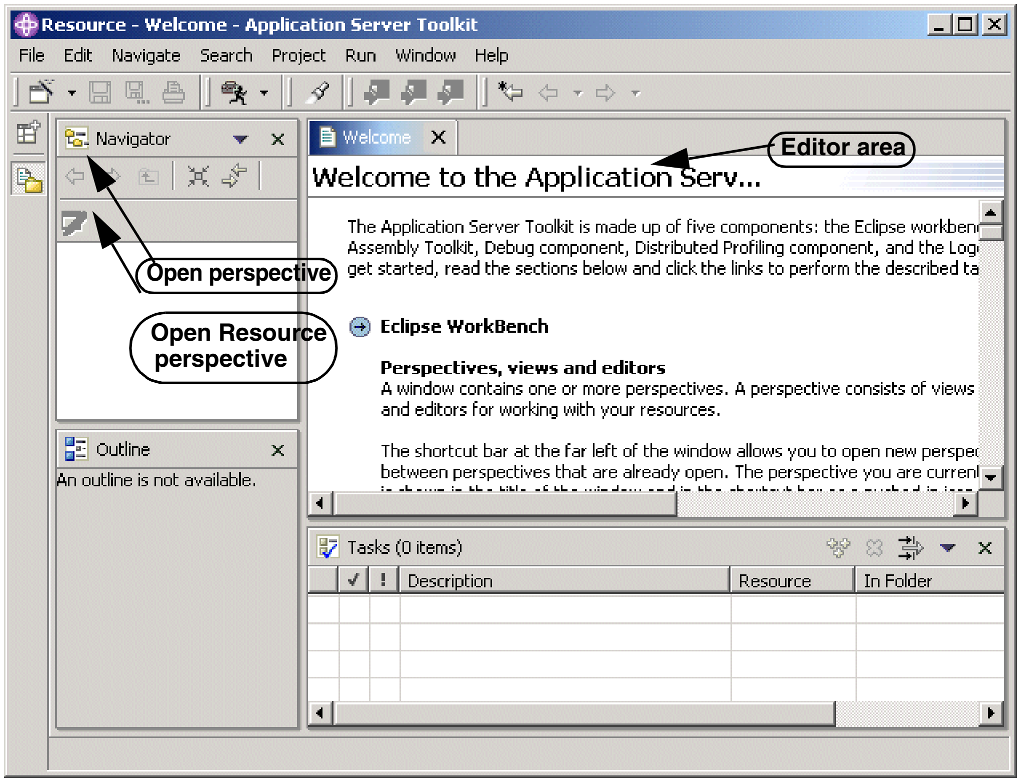
Since only one instance of the toolkit can be running using a particular workspace, you can also use the `-data` parameter to start a second instance.

Relative paths are interpreted relative to the directory from which the toolkit was started.

Tip: If, for some reason, you ever need to clean your workspace and start over from scratch, the quickest way would be to exit the toolkit and start it using a new workspace directory. You can then import the resources that you still require from the old workspace directory.

Navigating the Workbench

When you start the toolkit, the Workbench (Figure B-2) will open. The Workbench contains one or more perspectives, which are simply task-oriented groupings of views and editors. Depending on the task you are planning to perform (application assembly, debugging, etc.) you will select a perspective which is designed for the task. When you look at the Workbench, you are viewing it through one perspective.



Welcome to the Application Serv...

The Application Server Toolkit is made up of five components: the Eclipse workben
Assembly Toolkit, Debug component, Distributed Profiling component, and the Log
get started, read the sections below and click the links to perform the described ta

Eclipse WorkBench

Perspectives, views and editors
A window contains one or more perspectives. A perspective consists of views and editors for working with your resources.

The shortcut bar at the far left of the window allows you to open new perspe
between perspectives that are already open. The perspective you are curren

An outline is not available.

Figure B-2 Application Server Toolkit Workbench

At the far left of the window is a shortcut bar which allows you to open new perspectives and navigate between perspectives that are already open. The name of the active perspective is shown in the title of the window and its icon in the shortcut bar (left side) is a pushed button.

Using perspectives

The Workbench supports a role-based development model. It does so by providing several different perspectives on the same project. Perspectives provide a way to look at a project through different "glasses". Depending on the role you are in and/or the task you have to perform, you open a different perspective. A perspective defines an initial set and layout of views and editors for performing a particular set of activities, for example, EJB development, profiling, and so forth. You can change the layout and the preferences and save a perspective that you can have customized, so that you can open it again later.

Views

Views provide alternative presentations of resources or ways of navigating through the information in your Workbench. For example, the Navigator view displays projects and other resources that you are working with as a folder hierarchy. A view might appear by itself, or stacked with other views in a tabbed notebook arrangement. The content of views is synchronized, meaning for example that if you change a value in the Properties view, the Editor view is automatically updated to reflect the change.

Editors

When you open a file, it is opened with the editor associated with that file type. For example, an HTML editor is opened for .html, .htm and .jsp files while a Java editor is opened for .java and .jav files.

Editors that have been associated with specific file types open in the editor area of the Workbench. By default, editors are stacked in a notebook arrangement inside the editor area. You also have the option of tiling open files. However, if there is no associated editor for a resource, the toolkit will attempt to launch an external editor outside the Workbench.

You can view or change file associations by selecting **Window -> Preferences**. Select the **File Associations** option under the Workbench settings.

Switching perspectives

There are two ways to open another perspective. You can use the *Open a Perspective* icon in the top left corner of the Workbench working area and select the appropriate perspective from the list. Alternatively, you can click

Window -> Open Perspective and either select a perspective or click **Other** to bring up the Select Perspective dialog (Figure B-3).

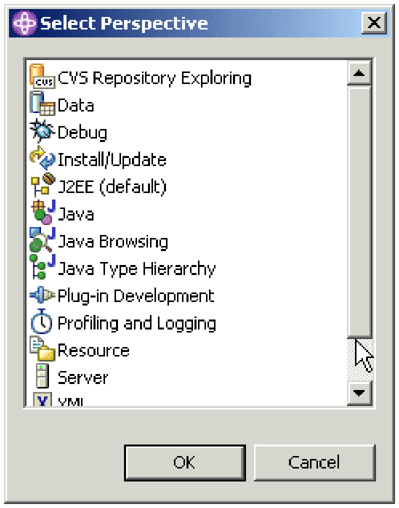


Figure B-3 Select Perspective dialog

Select the perspective you would like to open and click **OK**.

Tip: Having many perspectives open can slow down performance. To close a perspective, right-click that perspective's button on the shortcut bar and select **Close**.

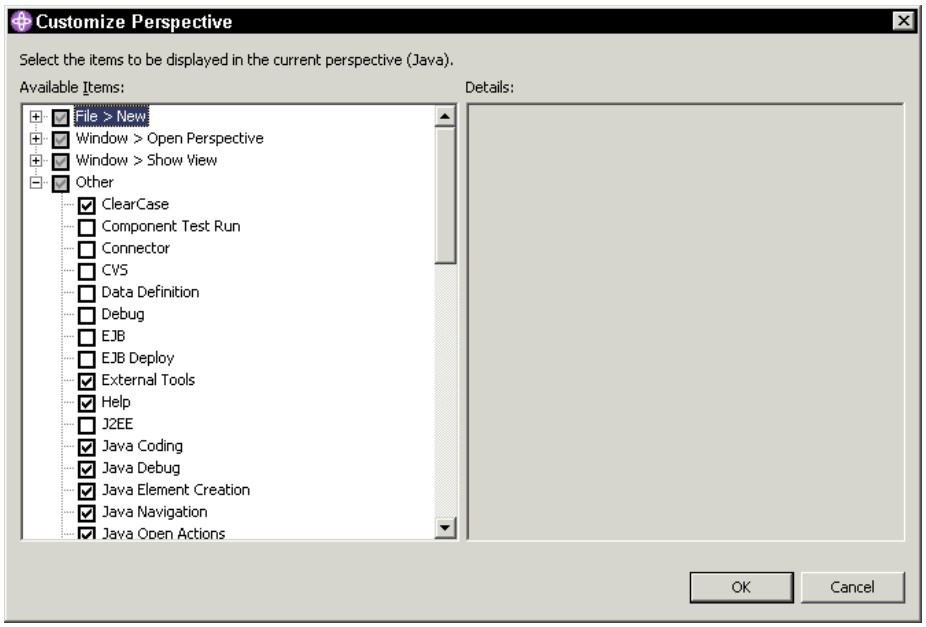
Organizing and customizing perspectives

The following features are available to help you in organizing perspectives:

- Open perspectives
- Customize perspectives
- Reset perspectives
- Save perspectives
- Close perspectives

These actions can be found in the Window menu.

To customize menu options available when you are in a perspective, click **Window -> Customize Perspective**. You can see the available options in Figure B-4 on page 885.


Customize Perspective

Select the items to be displayed in the current perspective (Java).

Available Items:

- ☑ File > New
- ☑ Window > Open Perspective
- ☑ Window > Show View
- ☑ Other
  - ☑ ClearCase
  - ☐ Component Test Run
  - ☐ Connector
  - ☐ CVS
  - ☐ Data Definition
  - ☐ Debug
  - ☐ EJB
  - ☐ EJB Deploy
  - ☑ External Tools
  - ☑ Help
  - ☐ J2EE
  - ☑ Java Coding
  - ☑ Java Debug
  - ☑ Java Element Creation
  - ☑ Java Navigation
  - ☑ Java Open Actions

Details:

Figure B-4 Customize perspective

In the dialog, you can use the check boxes to select which elements you want to see on the drop-down menus of the selected perspective. Items you do not select are still accessible by clicking the **Other** menu option. The following options can be customized:

- The **File -> New** menu
- The **Window -> Open Perspective** menu
- The **Window -> Show View** menu
- The **Other** option, which determines the icons that show up on the toolbar

You can also customize a perspective by adding, closing, moving, and resizing views. To add a new view to the perspective, simply click **Window -> Show View** and select the view you would like to add to the currently open perspective.

You can move a view to another pane by using drag and drop. To move a view, simply select its title bar, drag the view, and drop it on top of another view. Both views are now stacked and you can use the tabs at the bottom of the view to switch between them.

While you drag the view, the mouse cursor changes into a drop cursor. The drop cursor indicates what will happen when you release the view you are dragging:

⬇	The floating view appears below the view underneath the cursor.
⬅	The floating view appears to the left of the view underneath the cursor.
➡	The floating view appears to the right of the view underneath the cursor.
⬆	The floating view appears above the view underneath the cursor.
⧉	The floating view appears as a tab in the same pane as the view underneath the cursor. You can also drop the view on the perspective toolbar to make it a fast view.
⊘	You cannot dock the floating view at this point.

Once you have configured the perspective to your preferences, you can save it as your own perspective by selecting **Window -> Save Perspective As**. To restore the currently opened perspective to its original layout, select **Window -> Reset Perspective**.

Tip: You can double-click a view's title bar to maximize the view. Double-click it again to restore it to the original size. Alternatively, you can press **Ctrl-M** to maximize and restore the view.

Working with projects

A project is the top-level construct for organizing the different resources in the Workbench. It contains files as well as folders. The toolkit provides different project types for different tasks. Some projects are specific to toolkit functions while others (Figure B-5 on page 887) mirror the layout of J2EE enterprise application modules.

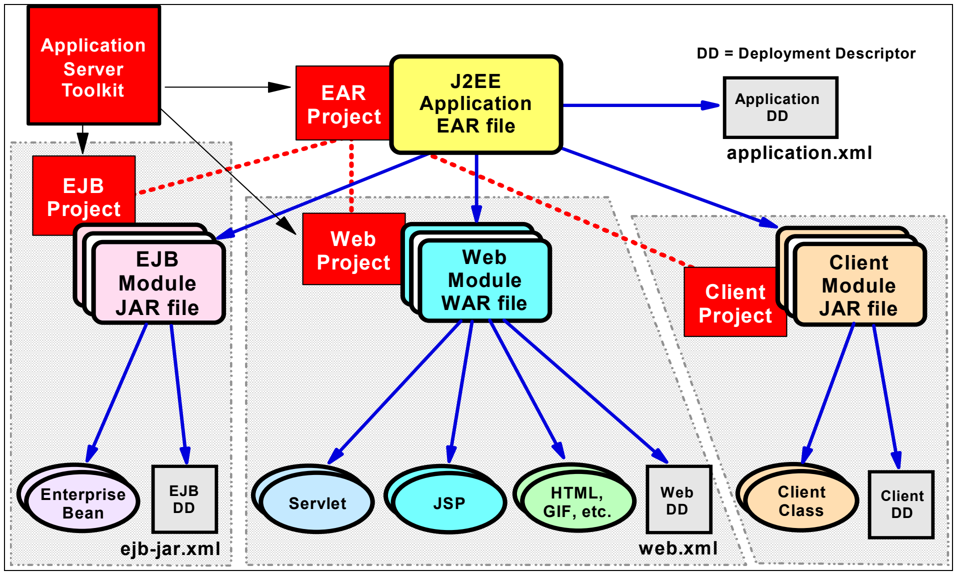


Figure B-5 J2EE projects

As projects are needed by the toolkit to perform specific functions, they will be created automatically.

Importing projects

When working with the toolkit, it is very likely that you will be importing J2EE enterprise applications or modules. You can import an enterprise application (.ear file), a Web module (.war file) or any other element of an application in one of two ways:

- You can use **File -> Import** to start the appropriate wizard. Select the file type to import and follow the wizard instructions.

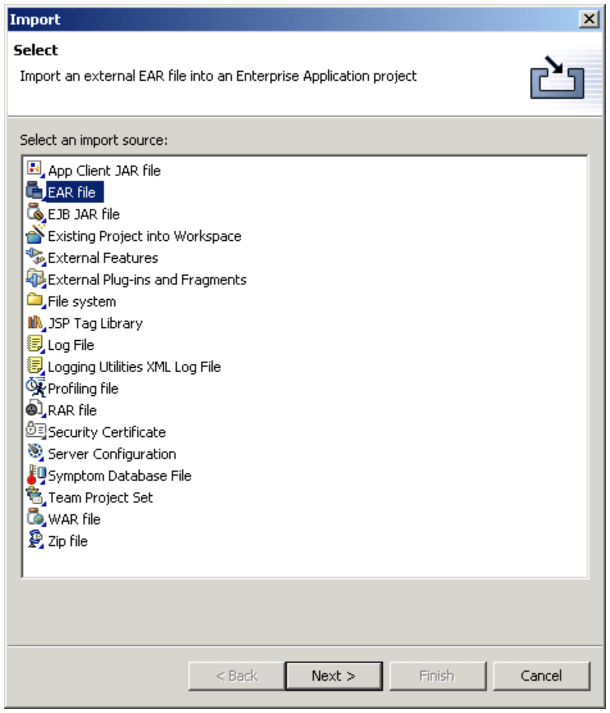


Figure B-6   Importing files

- In Windows, you can drag and drop files from a Windows file system into the toolkit. This will open the wizard needed to import the file.

Creating a new project

In some instances, you will need to create a project. For example, if you have imported Web and EJB modules but need to package them into an enterprise application, you can create an Enterprise Application project to add the modules to. When you export an Enterprise Application project, it is exported as an EAR file.

The Workbench provides wizards to create each specific type of project. You can invoke the appropriate wizard by selecting **File -> New -> Project** from the menu bar. The New Project wizard is shown in Figure B-7 on page 889.

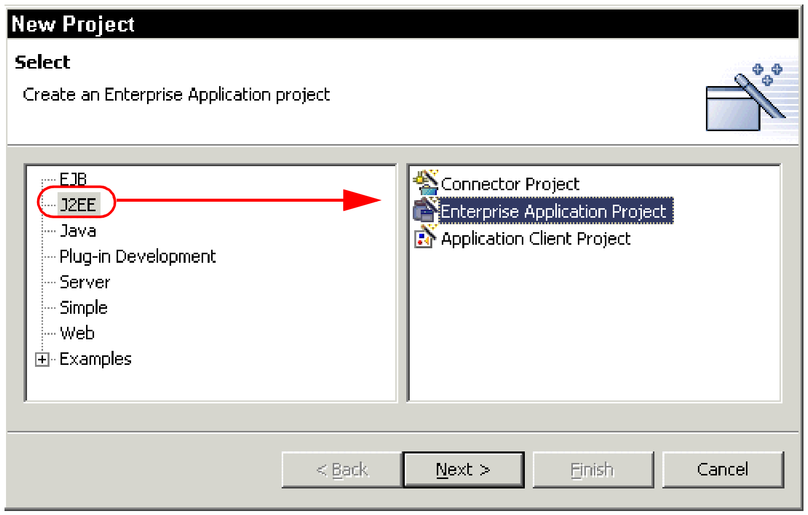


Figure B-7 New Project wizard

After the creation of a project, the appropriate perspective opens and displays a view of the project. Once you have created a project, you can make changes to the definition of a project by selecting the project in a Navigator view, and then selecting **Properties** from the context menu.

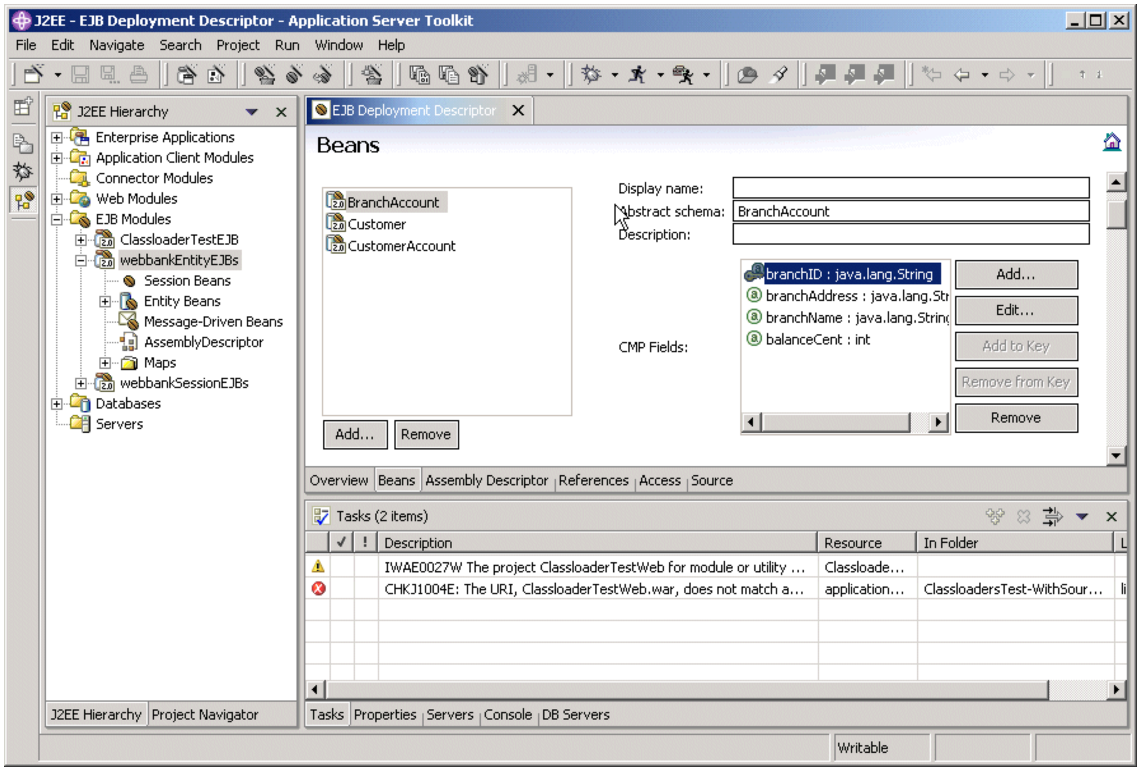
Assembly Toolkit

The Assembly Toolkit can be used to prepare enterprise application projects for deployment. The types of things you would use the Assembly Toolkit for are:

- Importing J2EE modules and creating enterprise applications and deployment descriptors. J2EE packaging includes J2EE 1.2 and J2EE 1.3 support.
- Importing existing EAR files and preparing them for deployment.
- Modifying and validating deployment descriptors.
- Setting runtime bindings.
- Generating EJB deployment and RMIC code.
- Exporting EAR files for deployment.
- Compiling Java code.
- Converting an EAR file from J2EE 1.2 to J2EE 1.3.
- Mapping EJB fields to database fields.
- Testing the deploy to WebSphere Application Server.

Using the Assembly Toolkit

To use the Assembly Toolkit, open the Application Server Toolkit and switch to the J2EE perspective. The J2EE perspective contains views that you would typically use when you work with enterprise application, EJB, Web, application client or connector projects.



✓	!	Description	Resource	In Folder	L
	⚠	IWAE0027W The project ClassloaderTestWeb for module or utility ...	Classloade...		
	⊗	CHKJ1004E: The URI, ClassloaderTestWeb.war, does not match a...	application...	ClassloadersTest-WithSour...	li

Figure B-8 Application Server Toolkit J2EE perspective

Let's take a quick look at some of the views in the J2EE perspective:

- To the left, you have two navigators that let you view the projects in the workspace: the *Project Navigator* and the *J2EE Hierarchy*. The Project Navigator provides a project and Java-centric view of the projects while the J2EE Hierarchy view provides a hierarchical view of the J2EE resources.

  Double-clicking an item in a navigator will open the item with the associated file editor in the editor area to the right. In Figure B-8, you can see that double-clicking the **BranchAccount** entity bean entry in the J2EE Hierarchy view has opened the EJB deployment descriptor with an editor designed to make it easy to work with deployment descriptors.

- The *Tasks view* lists automatically logged problems, warnings, or other information associated with the selected project. To address an item in the view, double-click the entry. The file which contains the problem will open to the location of the item that caused the entry.
- The *Properties view* provides a tabular view of the properties and associated values of objects in files you have open in an editor. For example, you can specify converters in the Properties view of the Mapping editor.

For an example of using the Assembly Toolkit to prepare an enterprise application for deployment, see:

- Chapter 13, “WebSphere specific packaging options” on page 617
- 14.2, “Setting application bindings” on page 669
- 14.3, “Generating deployment code” on page 675
- 14.4, “Deploying the application” on page 677

Debugging tools

The Application Server Toolkit offers a variety of debuggers that enable you to detect and diagnose errors in applications running locally or remotely. You can debug live server-side code, Java, and compiled languages. With the debuggers, you can control the execution of your programs by setting breakpoints, suspending execution, stepping through your code, and working with the contents of variables. The toolkit provides the ability to debug:

- J2EE applications: the WebSphere Application Server debug adapter allows you to debug Web objects that are running on WebSphere Application Server. These objects include EJBs, JSPs, and servlets. Through the use of the debug adapter, you can also debug server-side JavaScript.
- Java and compiled languages: the Java development tools debugger allows you to debug Java that is running locally or remotely. The compiled language debugger also provides remote and local debug capability, and allows you to debug languages such as C and C++.

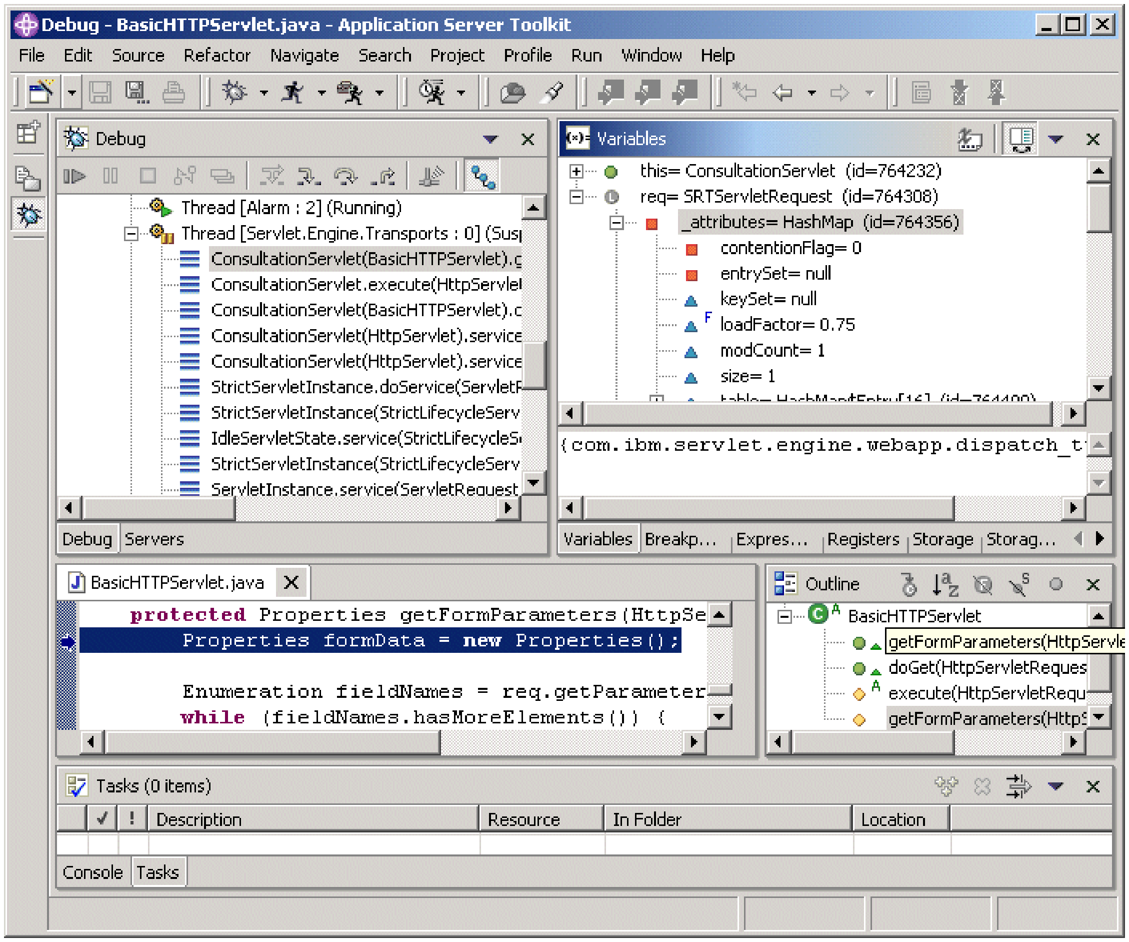
Debugging can be done through the use of breakpoints or by using step-by-step execution. Using breakpoints, you can designate stopping points in the program and examine the contents of variables. Using the step-by-step mode stops the application at entry to every Web object as it is loaded and allows you to choose whether to step into or over the code. Filters can be used to keep the application from stopping at every Web object.

Note that JavaScript and step-by-step debugging are only supported when debugging Web objects running on WebSphere Application Server V5.

For more details: For examples of using the Debug perspective, see *WebSphere Studio Application Developer Version 5 Programming Guide*, SG24-6957. Although it discusses debugging within the context of WebSphere Studio, the perspective is the same as the toolkit perspective. The primary difference is how you connect to the application server.

Debug perspective

The Debug perspective (Figure B-9) has views and functions that allow you to step through programs while inspecting, and sometimes controlling, the conditions which exist at each step.



```
Thread [Alarm : 2] (Running)
Thread [Servlet.Engine.Transports : 0] (Sus
    ConsultationServlet(BasicHTTPServlet).g
    ConsultationServlet.execute(HttpServle
    ConsultationServlet(BasicHTTPServlet).c
    ConsultationServlet(HttpServlet).service
    ConsultationServlet(HttpServlet).service
    StrictServletInstance.doService(ServletF
    StrictServletInstance(StrictLifecycleServ
    IdleServletState.service(StrictLifecycleS
    StrictServletInstance(StrictLifecycleServ
    ServletInstance.service(ServletRequest
```

```
this= ConsultationServlet  (id=764232)
req= SRTServletRequest  (id=764308)
    _attributes= HashMap  (id=764356)
        contentionFlag= 0
        entrySet= null
        keySet= null
        loadFactor= 0.75
        modCount= 1
        size= 1
```

```
(com.ibm.servlet.engine.webapp.dispatch_t
```

```
    protected Properties getFormParameters(HttpSe
        Properties formData = new Properties();

        Enumeration fieldNames = req.getParameter
        while (fieldNames.hasMoreElements()) {
```

```
BasicHTTPServlet
    getFormParameters(HttpServle
    doGet(HttpServletReques
    execute(HttpServletRequ
    getFormParameters(HttpS
```

	✓	!	Description	Resource	In Folder	Location

Figure B-9 Debugging applications in the Debug perspective

Some of the views you will use the most often are as follows.

- **Debug view**

  From the Debug view, which should now be displayed in the top left pane, you can use the functions available from its icon bar to control the execution of the application. The following icons are available:

  – Resume: runs the application to the next breakpoint
  – Suspend: suspends a running thread
  – Terminate: terminates a process
  – Disconnect: disconnects from the target when debugging remotely
  – Remove All Terminated Launches: removes terminated executions
  – Step Into: steps into the highlighted statement
  – Step Over: steps over the highlighted statement
  – Step Return: steps out of the current method
  – Step Debug: only for compiled languages (steps to next statement)
  – Step-by-step: toggles the step-by-step option

- **Breakpoints view**

  You can use the breakpoints view to display and manipulate the breakpoints that are currently set. You can open the properties (for example to set the hit count), remove the breakpoint, or open its source file.

- **Variables view**

  The Variables view displays the current values of the variables in the selected stack frame. If you want to test the code with some other value for any of these instance variables, you can change any one of them by selecting **Change Variable Value** from its context menu.

- **Expressions view and Display view**

  The Expressions and Display views are used to inspect variables and evaluate expressions in the context of the currently suspended thread. Both the Variables and Expressions view can be split into two panes by selecting **Show Detail Pane** from the context menu.

Approach to debugging

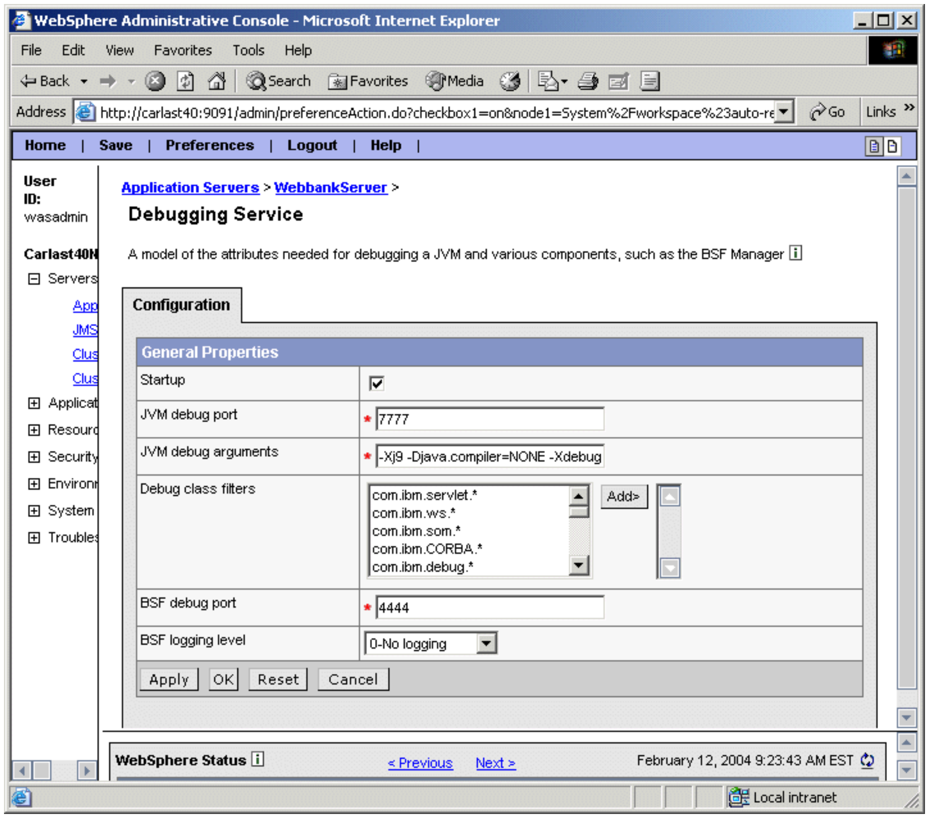
Let's take a quick look at the steps needed to debug an application on a remote WebSphere Application Server using the Debugger tools.

Preparing the application server

1. Deploy the application to the server.
2. Import the application into the Application Server Toolkit.
3. Open the source files and set breakpoints. Double-clicking the bar immediately to the left of the line of code will set the breakpoint. Save the file.

   Note: You do not need to deploy the files with the breakpoints to the server. The debugging function will work based on the source you have saved in the workspace.

4. Start the application server in debug mode. You do this from the WebSphere Administrative Console by selecting **Application Servers -> Servers** and the server name. Open the **Debugging Service** properties and select **Startup**.



Application Servers > WebbankServer >

Debugging Service

A model of the attributes needed for debugging a JVM and various components, such as the BSF Manager

Configuration

General Properties

Startup	✓
JVM debug port	7777
JVM debug arguments	-Xj9 -Djava.compiler=NONE -Xdebug
Debug class filters	com.ibm.servlet.* com.ibm.ws.* com.ibm.som.* com.ibm.CORBA.* com.ibm.debug.*
BSF debug port	4444
BSF logging level	0-No logging

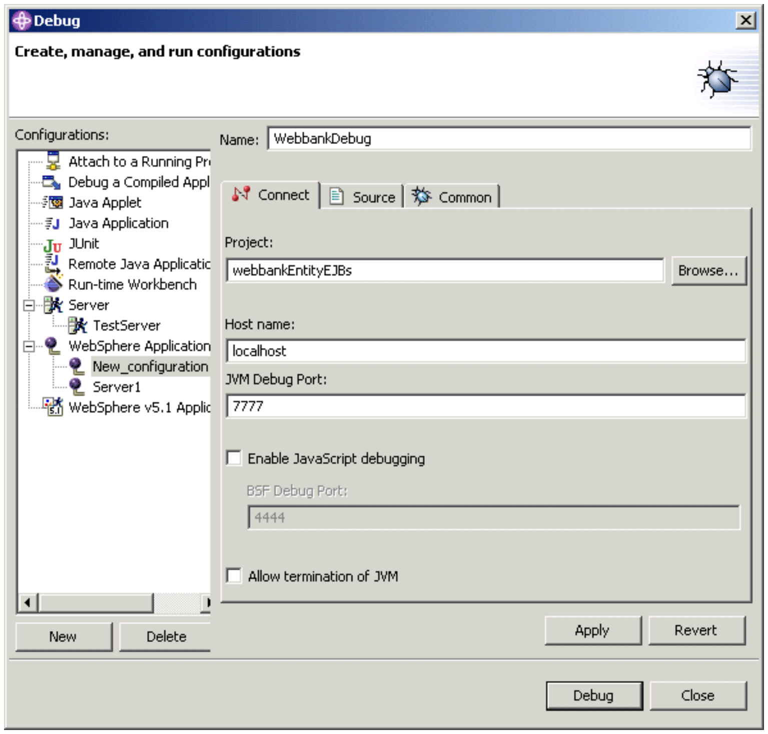
Figure B-10 Starting an application server in debug mode

5. Stop and restart the application server (running in debug mode will have a performance impact).

Preparing the toolkit

Now that the server is running in debug mode, you need to define the server to the toolkit and set the debug parameters.

1. Create a debug configuration in the toolkit. This is done in the toolkit Debug perspective by clicking the **Run -> Debug** option.
2. Select **WebSphere Application Server Debug** for the server type and click **New**.



Debug

Create, manage, and run configurations

Configurations:
- Attach to a Running Pr
- Debug a Compiled Appl
- Java Applet
- Java Application
- JUnit
- Remote Java Applicatic
- Run-time Workbench
- Server
  - TestServer
- WebSphere Application
  - New_configuration
  - Server1
- WebSphere v5.1 Applic

Name: WebbankDebug

Connect | Source | Common

Project: webbankEntityEJBs

Host name: localhost

JVM Debug Port: 7777

Enable JavaScript debugging: unchecked

BSF Debug Port: 4444

Allow termination of JVM: unchecked

Figure B-11 Create a launch configuration

Enter the settings needed to match your environment to the runtime. Note that the JVM debug port is the same as shown in the server's Debugging Service properties in Figure B-10 on page 894.

Click the **Source** tab and make sure that the project source location is included in the lookup path. If the Debugger can find the source, it will be displayed as you step through the code.

3. Click **Debug**.

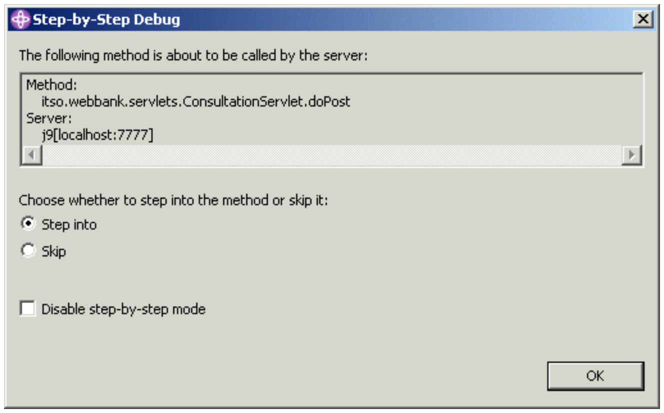
   If you want to use the step-by-step mode, make sure it is enabled.

Debugging the application

Now it is time to start the application and use the debugging tools to determine what is happening during execution.

1. Open a Web browser and invoke the application.

   If you are using breakpoints, the application will stop at each breakpoint. In step-by-step mode, a window will open as each object is entered. It will provide the option of executing the code without stepping through, or to step through the code.



The following method is about to be called by the server:

```
Method:
   itso.webbank.servlets.ConsultationServlet.doPost
Server:
   j9[localhost:7777]
```

Choose whether to step into the method or skip it:

[✓ Step into] Skip

☐ Disable step-by-step mode

Figure B-12 Step-by-step mode

2. Each time the application stops, either for a breakpoint or because you have chosen to step into the code, you have the opportunity to examine the state of the application. To resume, click the **Resume** icon.
3. When you are done, disconnect by highlighting the launch configuration in the Debug view and clicking the **Disconnect** icon.

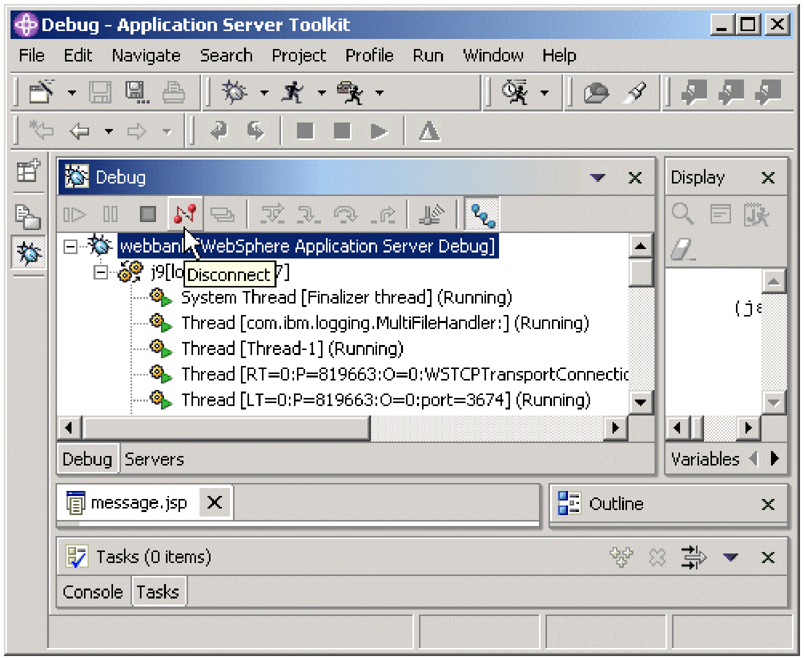

Debug - Application Server Toolkit
File Edit Navigate Search Project Profile Run Window Help
Debug
webbank [WebSphere Application Server Debug]
j9[lo Disconnect 7]
System Thread [Finalizer thread] (Running)
Thread [com.ibm.logging.MultiFileHandler:] (Running)
Thread [Thread-1] (Running)
Thread [RT=0:P=819663:O=0:WSTCPTransportConnectic
Thread [LT=0:P=819663:O=0:port=3674] (Running)
Debug Servers
Display
Variables
message.jsp
Outline
Tasks (0 items)
Console Tasks

Figure B-13 Disconnect

Application profiling

The performance of an application must be a focus area throughout the project cycle. Traditionally, performance testing has occurred late in the cycle, towards the end of the testing phase, or as part of the deployment process. However, it is a good idea to ensure that applications perform adequately from the beginning. To support performance monitoring at each stage of development all the way through to deployment, the WebSphere Studio tools and the Application Server Toolkit include facilities for profiling and measuring performance of Web applications.

For more information: For information on application profiling, see I*BM WebSphere V5.1 Performance, Scalability, and High Availability,* SG24-6198-01.

The profiling tools can be used to collect and display data relative to the runtime behavior of the application and to provide insight into how the Java Virtual Machine (JVM) is executing the application. The tools work in conjunction with the Agent Controller.

The profiling tools can be used to analyze object creation and garbage collection behavior, execution sequences, thread interaction, and object references. This will provide an indication of which parts of the application tuning activities you should focus on.

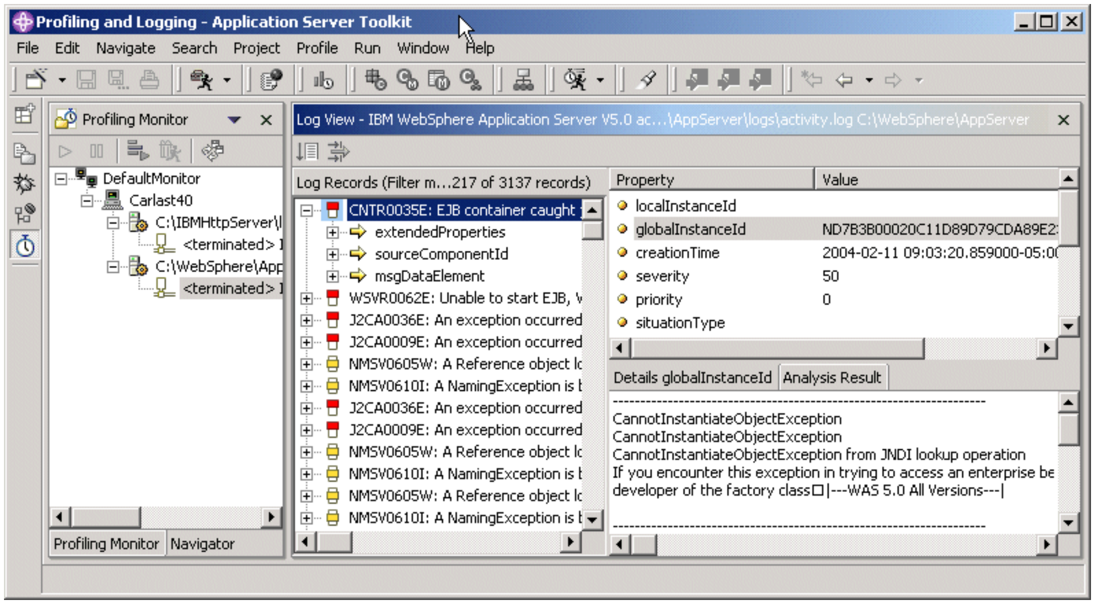
The tools can display profiling data in a number of graphical and statistical (tabular) views that highlight different aspects of the application's runtime behavior. The information that is provided by the profiling tools includes:

- Graphical profiling views:
  - Execution Flow view and table
  - Method Invocations view and table
- Sequence diagrams:
  - Object interactions
  - Class interactions
  - Thread interactions
  - Process interactions
  - Host interactions
- Statistical profiling views:
  - Package statistics
  - Class statistics
  - Method statistics
  - Instance statistics

Log Analyzer

The Log Analyzer plug-in is similar to the Log Analyzer tool that is installed with WebSphere Application Server, but with a Workbench look and feel. Web server, WebSphere Application Server, and DB2 logs can be imported into the workspace and then analyzed by matching them to one or more imported symptom databases. In addition, log entries from multiple logs can be correlated into a time or event sequence.

The Profiling and Logging perspective is used for the Log Analyzer functions.


Profiling and Logging - Application Server Toolkit
File Edit Navigate Search Project Profile Run Window Help
Profiling Monitor
DefaultMonitor
Carlast40
C:\IBMHttpServer\l
<terminated> I
C:\WebSphere\App
<terminated> I
Profiling Monitor
Navigator
Log View - IBM WebSphere Application Server V5.0 ac...\AppServer\logs\activity.log C:\WebSphere\AppServer
Log Records (Filter m...217 of 3137 records)
Details globalInstanceId
Analysis Result

```
CNTR0035E: EJB container caught
    extendedProperties
    sourceComponentId
    msgDataElement
WSVR0062E: Unable to start EJB, \
J2CA0036E: An exception occurred
J2CA0009E: An exception occurred
NMSV0605W: A Reference object lo
NMSV0610I: A NamingException is t
J2CA0036E: An exception occurred
J2CA0009E: An exception occurred
NMSV0605W: A Reference object lo
NMSV0610I: A NamingException is t
NMSV0605W: A Reference object lo
NMSV0610I: A NamingException is t
```

Property	Value
localInstanceId	
globalInstanceId	ND7B3B00020C11D89D79CDA89E2:
creationTime	2004-02-11 09:03:20.859000-05:0(
severity	50
priority	0
situationType	

```
-----------------------------------------------------------------------
CannotInstantiateObjectException
CannotInstantiateObjectException
CannotInstantiateObjectException from JNDI lookup operation
If you encounter this exception in trying to access an enterprise be
developer of the factory class□|---WAS 5.0 All Versions---|

-----------------------------------------------------------------------
```

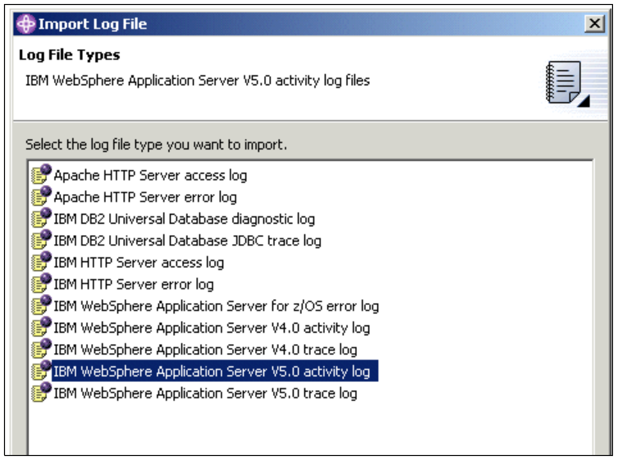
Figure B-14 Profiling and Logging perspective

- The Profiling Monitor view shows the log files that have been imported.
- Selecting a log file will display its entries in the Log Records pane of the Log view. You can filter this view to cut down on the number of records displayed.
- Selecting a log record will display its properties in the Property and Value panes. Selecting a property will show detailed information in the Details pane.
- Each log record can be analyzed against a symptom database. The results of the analysis are shown in the Analysis Result pane.

Importing log files

The first step is to import the log files.

1. Open the Profiling and Logging perspective.
2. Select **File -> Import**.
3. Select **Log File** and click **Next**.
4. Select the type of file you need from the options and click **Next**.


Import Log File

Log File Types

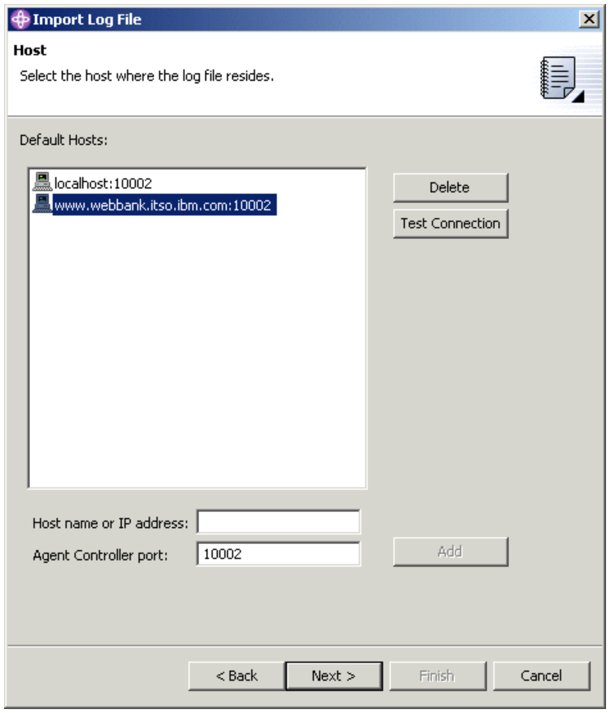
IBM WebSphere Application Server V5.0 activity log files

Select the log file type you want to import.

- Apache HTTP Server access log
- Apache HTTP Server error log
- IBM DB2 Universal Database diagnostic log
- IBM DB2 Universal Database JDBC trace log
- IBM HTTP Server access log
- IBM HTTP Server error log
- IBM WebSphere Application Server for z/OS error log
- IBM WebSphere Application Server V4.0 activity log
- IBM WebSphere Application Server V4.0 trace log
- IBM WebSphere Application Server V5.0 activity log
- IBM WebSphere Application Server V5.0 trace log

Figure B-15 Importing log files

5. The next panel allows you to specify the host where the log file is located and to test the oonnection to the Agent Controller running on that host. Note that the port for the Agent Controller defaults to 10002. If you have conflicts with this port, you can change it in the configuration file for the Agent Controller, located at <agent_install>/config/serviceconfig.xml.



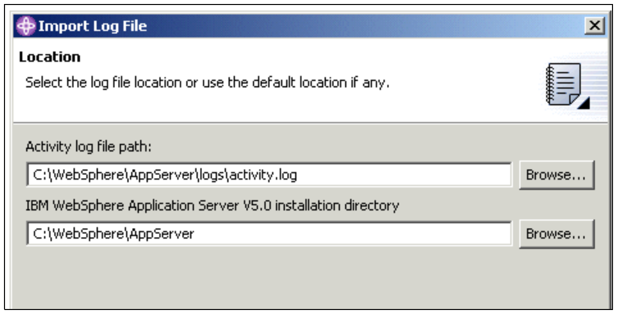
localhost:10002
www.webbank.itso.ibm.com:10002

Host name or IP address:
Agent Controller port: 10002

Figure B-16 Select the remote host location

If you need to add a new host location, type in the host name, verify that the Agent Controller port is correct and click **Add**. Select the host in the Default Hosts window and click **Test Connection** to make sure the Agent Controller is running and can be reached using the host name and port you specified. When you are satisfied that the connection is working, click **Next**.

6. Enter the location for the log file. Depending on the log type you are importing, you may need to enter other information. If the host entered was the local host, you will be able to browse to the location.



Location

Select the log file location or use the default location if any.

Activity log file path:

C:\WebSphere\AppServer\logs\activity.log

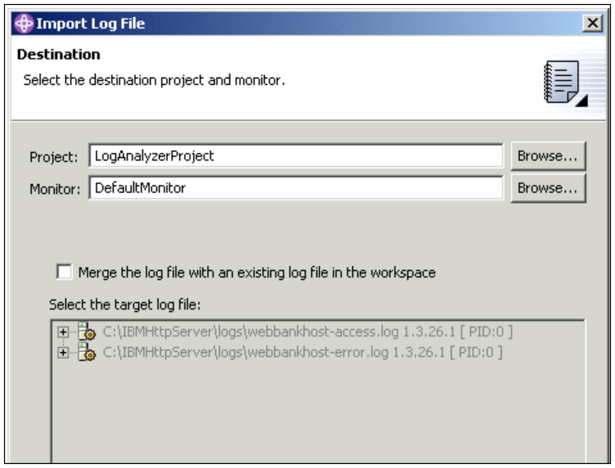
IBM WebSphere Application Server V5.0 installation directory

C:\WebSphere\AppServer

Figure B-17 Enter the location of the log file

Click **Next**.

7. Select the destination project and monitor. The default for these fields is shown in Figure B-18.



Destination

Select the destination project and monitor.

Project: LogAnalyzerProject

Monitor: DefaultMonitor

☐ Merge the log file with an existing log file in the workspace

Select the target log file:

```
C:\IBMHttpServer\logs\webbankhost-access.log 1.3.26.1 [ PID:0 ]
C:\IBMHttpServer\logs\webbankhost-error.log 1.3.26.1 [ PID:0 ]
```

Figure B-18 Select the toolkit projects to use

You can opt to merge the new file with an existing file. If you select that option, you will be asked whether you want to append the log file to the existing file, or replace the existing file. If you are importing several log files which you want to correlate together and you are not using a correlation plug-in, you will need to merge the logs together.

Click **Next**.

8. Click **Finish**. The file will be imported and you will see it in the Profile Monitor view.

Analyzing entries

Symptom databases are XML files used to match text patterns in the log with solution text. You can download a symptom database from IBM support from within the tool, or you can create your own custom database.

Importing the WebSphere symptom database

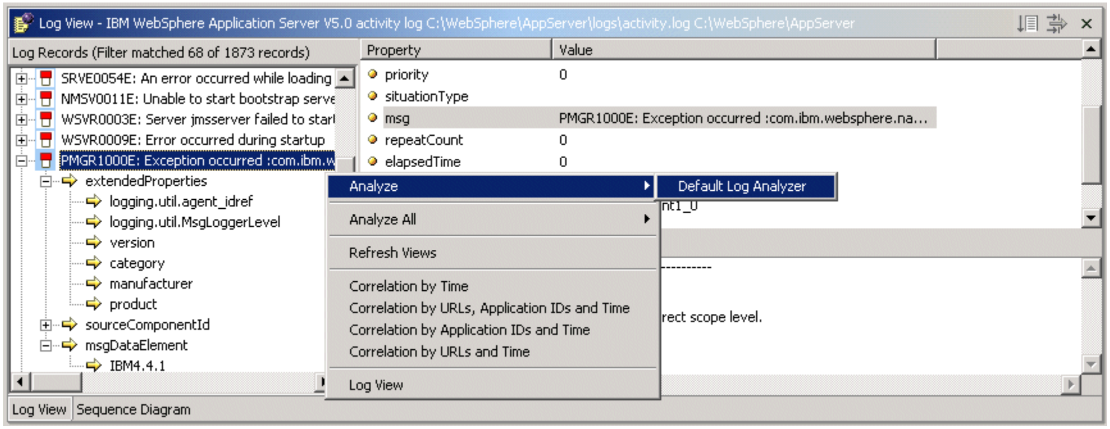
To import a database, follow these steps:

1. Select **File -> Import -> Symptom Database File**. Click **Next**.
2. The next window lets you select the file to import. You can download the standard WebSphere Application Server symptom file by selecting **Remote host** and then **WebSphere Application Server Advanced Edition** in the drop-down menu. Enter the project and file names (they default for you) and click **Finish**.

Analyzing records

To analyze a record against the symptom database, do this:

1. Select a log entry.
2. Right-click and select **Analyze -> Default Log Analyzer**.



Log View - IBM WebSphere Application Server V5.0 activity log C:\WebSphere\AppServer\logs\activity.log C:\WebSphere\AppServer
Log Records (Filter matched 68 of 1873 records)
Property
Value
Analyze
Default Log Analyzer
Analyze All
Refresh Views
Correlation by Time
Correlation by URLs, Application IDs and Time
Correlation by Application IDs and Time
Correlation by URLs and Time
Log View
Log View
Sequence Diagram

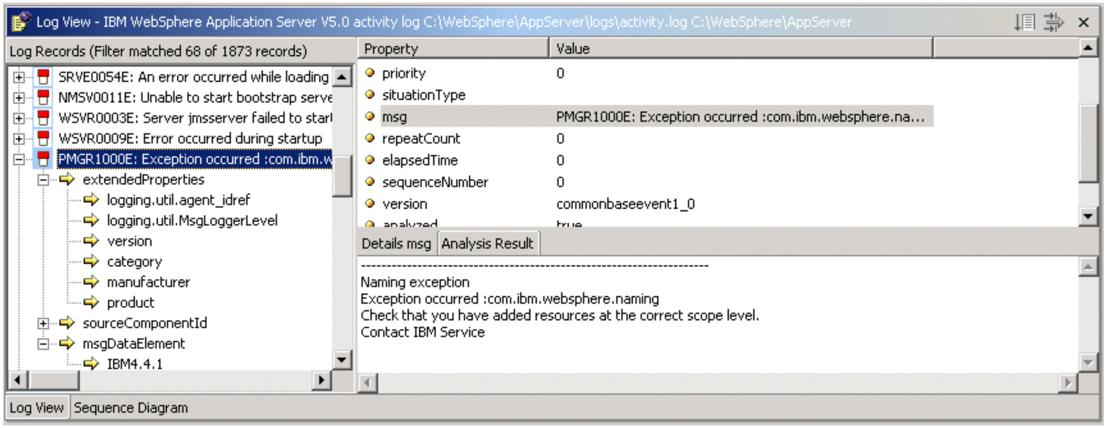
```
SRVE0054E: An error occurred while loading
NMSV0011E: Unable to start bootstrap serve
WSVR0003E: Server jmsserver failed to start
WSVR0009E: Error occurred during startup
PMGR1000E: Exception occurred :com.ibm.w
  extendedProperties
    logging.util.agent_idref
    logging.util.MsgLoggerLevel
    version
    category
    manufacturer
    product
  sourceComponentId
  msgDataElement
    IBM4.4.1
```

Property	Value
priority	0
situationType	
msg	PMGR1000E: Exception occurred :com.ibm.websphere.na...
repeatCount	0
elapsedTime	0

```
nt1_0
----------
rect scope level.
```

Figure B-19 Analyzing records

3. Switch to the Analysis Results pane to see the results.


Log View - IBM WebSphere Application Server V5.0 activity log C:\WebSphere\AppServer\logs\activity.log C:\WebSphere\AppServer
Log Records (Filter matched 68 of 1873 records)
Property
Value
Details msg
Analysis Result
Log View
Sequence Diagram

```
SRVE0054E: An error occurred while loading
NMSV0011E: Unable to start bootstrap serve
WSVR0003E: Server jmsserver failed to start
WSVR0009E: Error occurred during startup
PMGR1000E: Exception occurred :com.ibm.w
  extendedProperties
    logging.util.agent_idref
    logging.util.MsgLoggerLevel
    version
    category
    manufacturer
    product
  sourceComponentId
  msgDataElement
    IBM4.4.1
```

Property	Value
priority	0
situationType	
msg	PMGR1000E: Exception occurred :com.ibm.websphere.na...
repeatCount	0
elapsedTime	0
sequenceNumber	0
version	commonbaseevent1_0
analyzed	true

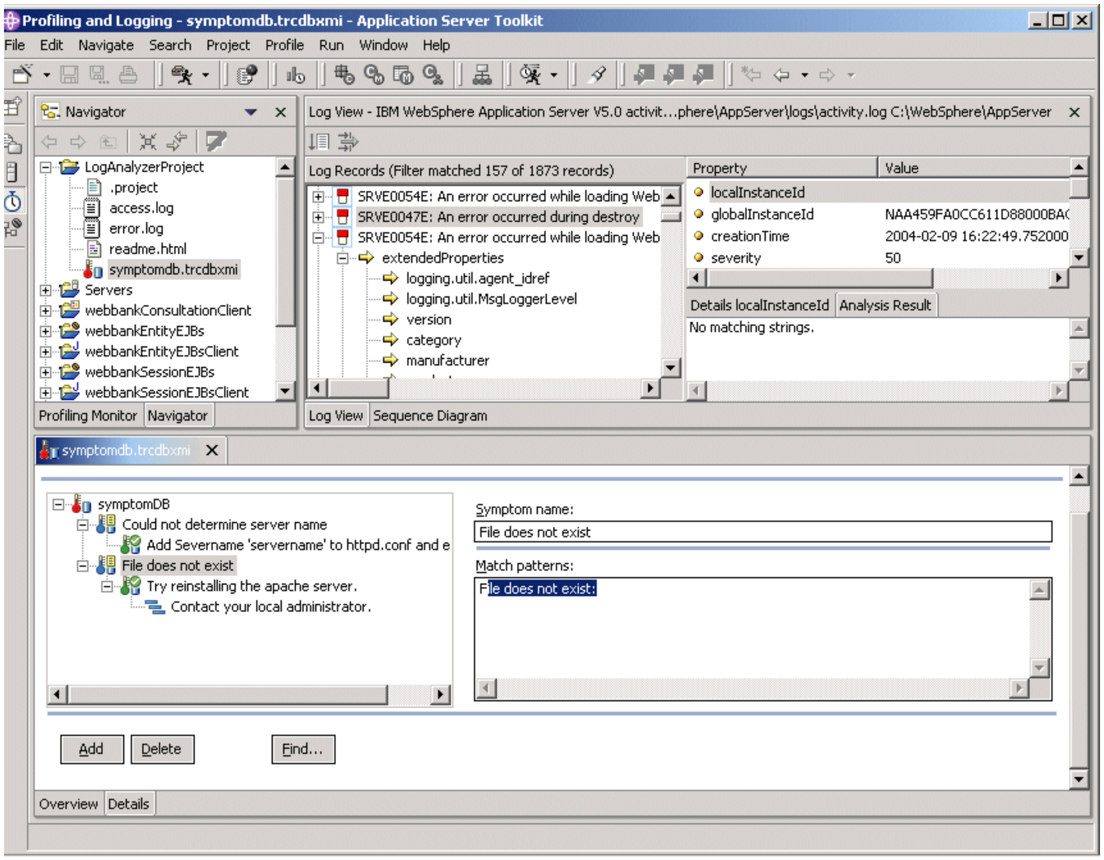
```
-----------------------------------------------------------------------
Naming exception
Exception occurred :com.ibm.websphere.naming
Check that you have added resources at the correct scope level.
Contact IBM Service
```

Figure B-20 Analysis results

Customizing the symptom database

To customize the symptom database, follow these steps:

1. Find the symptom file in the Profile Monitor view and double-click it to open it. The default Log Analyzer symptom database can be found in the LogAnalyzer project. The file is symptomdb.trcdbxmi. The symptom file will open in an editor at the bottom of the view.
2. Open the Details view to see the existing symptoms. You can select a symptom to see the text pattern used to match a log record, and the solution.


Profiling and Logging - symptomdb.trcdbxmi - Application Server Toolkit
File Edit Navigate Search Project Profile Run Window Help
Navigator
Log View - IBM WebSphere Application Server V5.0 activit...phere\AppServer\logs\activity.log C:\WebSphere\AppServer
Profiling Monitor
Navigator
Log View
Sequence Diagram
Overview
Details
Add
Delete
Find...

LogAnalyzerProject
.project
access.log
error.log
readme.html
symptomdb.trcdbxmi
Servers
webbankConsultationClient
webbankEntityEJBs
webbankEntityEJBsClient
webbankSessionEJBs
webbankSessionEJBsClient

Log Records (Filter matched 157 of 1873 records)

```
SRVE0054E: An error occurred while loading Web
SRVE0047E: An error occurred during destroy
SRVE0054E: An error occurred while loading Web
  extendedProperties
    logging.util.agent_idref
    logging.util.MsgLoggerLevel
    version
    category
    manufacturer
```

Property	Value
localInstanceId	
globalInstanceId	NAA459FA0CC611D88000BA(
creationTime	2004-02-09 16:22:49.752000
severity	50

Details localInstanceId | Analysis Result

No matching strings.

symptomdb.trcdbxmi

```
symptomDB
  Could not determine server name
    Add Severname 'servername' to httpd.conf and e
  File does not exist
    Try reinstalling the apache server.
      Contact your local administrator.
```

Symptom name: File does not exist

Match patterns: File does not exist:

Figure B-21 Viewing the symptom database

3. You can add symptoms or directives (solutions) by using the context menu options for the entries in the database, or by clicking the **Add** and **Delete** buttons at the bottom.

   The item that will be added depends on the entry you have selected. If you select the database and click **Add**, a new symptom will be added. If you select a symptom and click **Add**, a new solution will be added. If you select a solution, a new directive will be added.

4. Update the fields for the new entry and save the database file.

Tip: Rather than updating the default symptom database, it is better to create your own. To create the database, use the **File -> New -> Symptom Database** wizard. Make sure that it is enabled for your log analysis by checking the **Window -> Preferences -> Profiling and Logging -> Log View -> Symptom Database** settings.

Correlating databases

When performing problem determination across products, it is often difficult to see the whole picture of an event. Each product does its own logging and tracing, in its own format. The Log Analyzer in the toolkit allows you to import log and trace files from the following products and correlate the activities in them, providing one comprehensive picture of an event.

- IBM WebSphere Application Server V4.0 and V5.0 activity log files and trace log files
- IBM HTTP Server V1.3.19.3 to V2.0 access log files and error log files
- Apache HTTP Server V1.3.20, V1.3.26 and V2.0 access log files and error log files
- IBM DB2 UDB V8.1.1 and V8.1.2 diagnostic logs
- IBM DB2 UDB V8.1 fix pack 2 JDBC trace logs
- Logging Utilities XML log file

As part of the base implementation, correlation can only be done within a single file. To correlate multiple files using this implementation, you will need to merge the log files into a single file when you import them. The Log Analyzer also provides the ability to analyze and correlate events in multiple logs, using algorithms that are imported and applied as a set of correlation plug-ins.

The base implementation of the Log Analyzer provides several correlation methods for events in a single product log:

- Sequential correlation: the ability to sort the events in a log by various fields contained in the events (for example, the time stamp).
- Associative correlation: the ability to filter the events displayed in a log by the values in various fields contained in the events (for example, thread ID).

The Log Analyzer provides the option to display correlations in two ways:

- Log interaction diagrams, which display interactions among log events (records) that occur during the execution of an application.

- Log thread interaction diagrams, which display interactions among log events (records) that occur on different threads, participating in the execution of an application.

Log interactions

To display log interactions:

1. Import the log files to correlate, merging them into a single file.
2. Select the file and then select **Open With -> Log Interactions** from the context menu.

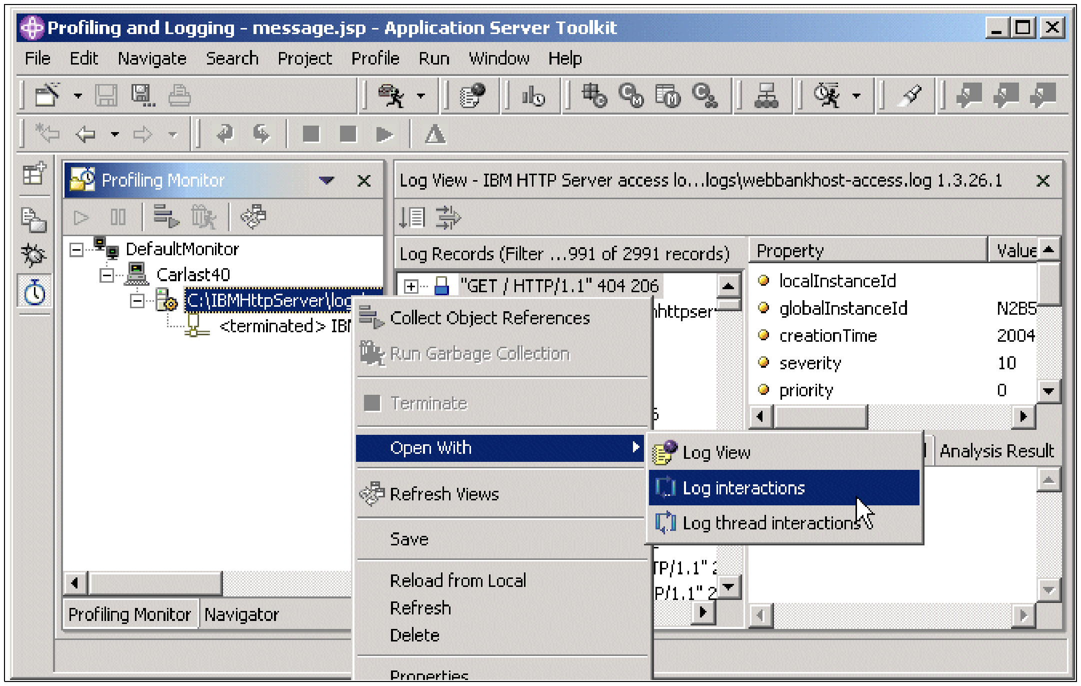
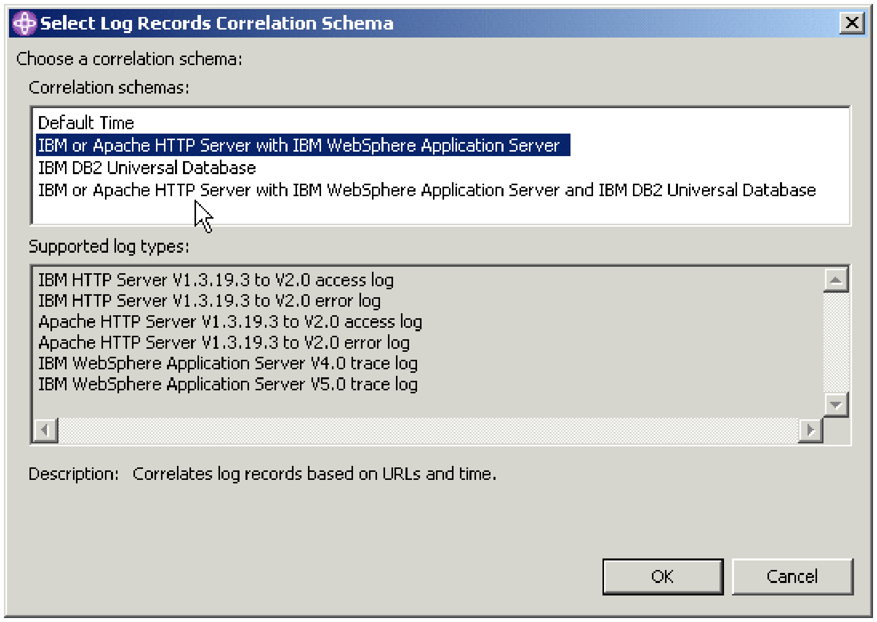


Figure B-22 Correlating logs

3. Select the type of correlation you want to use and click **OK**. Selecting an entry will display the details of how records will be correlated at the bottom of the panel.


Select Log Records Correlation Schema
Choose a correlation schema:
Correlation schemas:

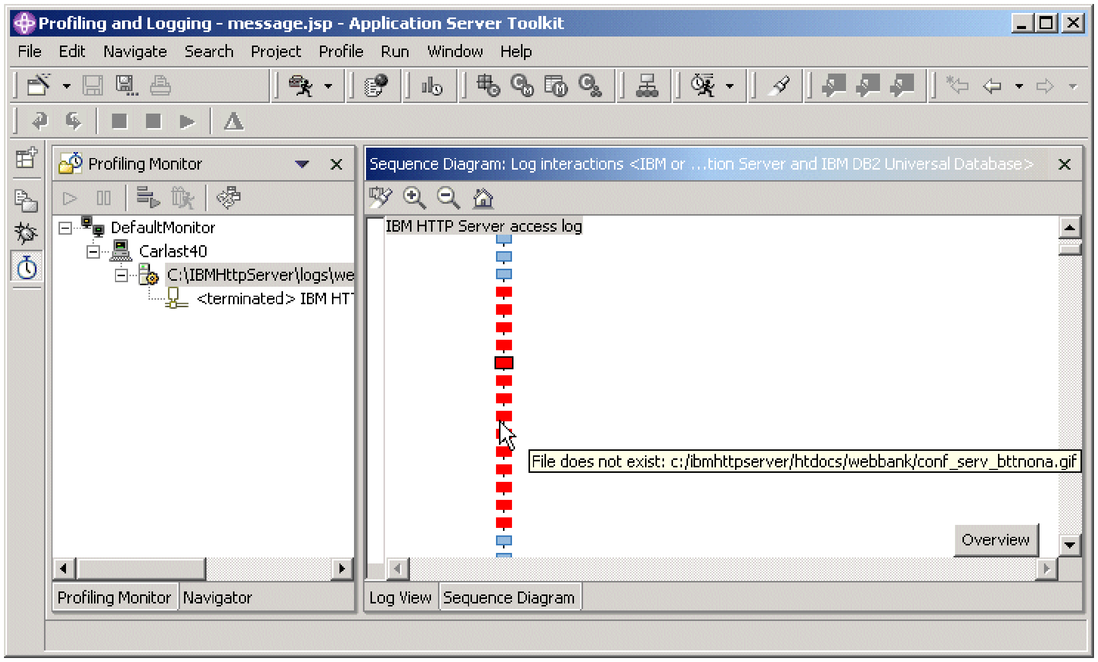
Default Time
IBM or Apache HTTP Server with IBM WebSphere Application Server
IBM DB2 Universal Database
IBM or Apache HTTP Server with IBM WebSphere Application Server and IBM DB2 Universal Database

IBM HTTP Server V1.3.19.3 to V2.0 access log
IBM HTTP Server V1.3.19.3 to V2.0 error log
Apache HTTP Server V1.3.19.3 to V2.0 access log
Apache HTTP Server V1.3.19.3 to V2.0 error log
IBM WebSphere Application Server V4.0 trace log
IBM WebSphere Application Server V5.0 trace log

Description: Correlates log records based on URLs and time.

Figure B-23 Select the correlation schema

All log files supporting this type of correlation will participate in correlation and the Sequence Diagram view will open.


Profiling and Logging - message.jsp - Application Server Toolkit
File Edit Navigate Search Project Profile Run Window Help
Profiling Monitor
DefaultMonitor
Carlast40
C:\IBMHttpServer\logs\we
<terminated> IBM HT
Sequence Diagram: Log interactions <IBM or ...tion Server and IBM DB2 Universal Database>
IBM HTTP Server access log
Overview
Profiling Monitor
Navigator
Log View
Sequence Diagram

File does not exist: c:/ibmhttpserver/htdocs/webbank/conf_serv_bttnona.gif

Figure B-24 Log sequence diagrams

Holding the mouse over an element in the diagram will display a pop-up for the entry.

Log thread interactions

To display log thread interactions:

1. Import into the log files to correlate, merging them into a single file.
2. Select the file and then select **Open With -> Log Thread Interactions** from the context menu.

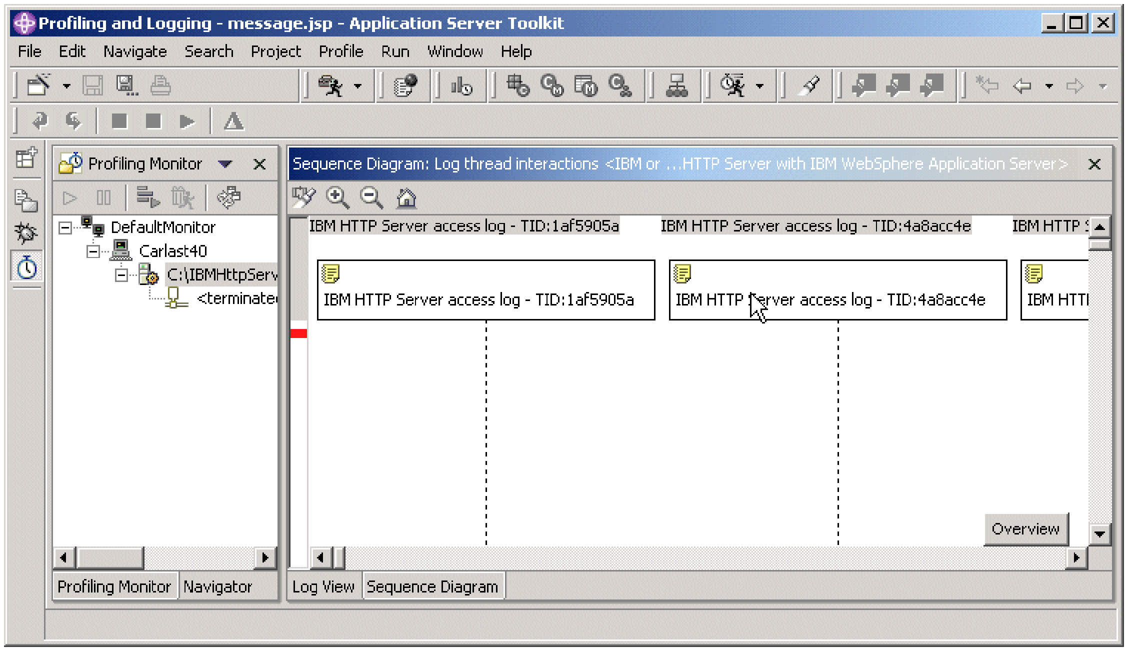

Profiling and Logging - message.jsp - Application Server Toolkit
File Edit Navigate Search Project Profile Run Window Help
Profiling Monitor
DefaultMonitor
Carlast40
C:\IBMHttpServ
<terminate
Sequence Diagram: Log thread interactions <IBM or ...HTTP Server with IBM WebSphere Application Server>
IBM HTTP Server access log - TID:1af5905a
IBM HTTP Server access log - TID:4a8acc4e
Overview
Profiling Monitor
Navigator
Log View
Sequence Diagram

Figure B-25 Log thread interactions

C

Additional troubleshooting information

The following is an illustration of creating a creating a servlet to dump session data for debugging purposes.

Compile the following SessionContextMonitor.java file.

Example: C-1 SessionContextMonitor.java

```
package com.ibm.ws.webcontainer.httpsession;
import java.util.*;
public class SessionContextMonitor
{
    public SessionContextMonitor(){}
    public static Enumeration getScrSessionContexts(){
        return SessionContextRegistry.getScrSessionContexts();
    }
    public static Enumeration tableKeys(SessionContext sc){
        return sc.tableKeys();
    }
    public static Object tableGet(SessionContext sc, Object key) {
        return sc.tableGet(key);
    }
}
```

Compiling steps:

- For Windows systems:

```
C:\> set WAS_HOME=C:\WebSphere\AppServer
C:\> set JAVA_HOME=%WAS_HOME%\java
C:\> set PATH=%JAVA_HOME%\bin;%PATH%

C:\> mkdir work
C:\> cd work
C:\work> javac -classpath %WAS_HOME%\lib\httpsession.jar;%WAS_HOME%\lib\web
container.jar -d .   c:\downloads\SessoinContextMonitor.java
c:\work> jar -cvf httpsessionmon5.jar com
...
c:\work> copy httpsessionmon5.jar %WAS_HOME%\lib\
```

For UNIX systems:

```
# ksh
$ export WAS_HOME=/usr/WebSphere/AppServer
$ export JAVA_HOME=$WAS_HOME/java
$ export PATH=$JAVA_HOME/bin:$PATH

$ md /home/work
$ cd /home/work
/home/work/$ javac -classpath $WAS_HOME/lib/httpsession.jar:$WAS_HOME/lib/
webcontainer.jar -d .   /tmp/SessionContextMonitor.java
/home/work/$ jar -cvf httpsessionmon5.jar com
...
/home/work/$ cp httpsessionmon5.jar $WAS_HOME/lib/
```

Restart the application server and put the following sessoinmon5.jsp file into a directory you can access via a Web browser.

Example: C-2   sessionmon5.jsp

```
<%@ page session="false" buffer="none" contentType="text/html" %>
<%@ page import="java.io.*,java.util.*,com.ibm.ws.webcontainer.httpsession.*"
%>
<html><head><title>IBM WebSphere 5.x Session Monitor</title></head>
<body>
<table width=600><tr><td>
<center><h3>IBM WebSphere 5.x Session Monitor</h3></center>
</td></tr></table>
<% String ip =java.net.InetAddress.getLocalHost().getHostAddress();
   java.text.SimpleDateFormat df =
      new java.text.SimpleDateFormat("yyyy.MM.dd HH:mm:ss");
   String date = df.format(new java.util.Date());
%>
OS : <%= System.getProperty("os.name")+System.getProperty("os.version")%><br>
IP Address : <%= ip %><br>
```

```
Date : <%= date %><br><br>
JDK fullversion : <%= System.getProperty("java.fullversion") %><br>
user.language : <%= System.getProperty("user.language") %><br>
user.regiong : <%= System.getProperty("user.region") %><br>
user.timezone : <%= System.getProperty("user.timezone") %><br>
file.encoding : <%= System.getProperty("file.encoding") %><br>
<%
try{
   Enumeration sessionContexts =
        SessionContextMonitor.getScrSessionContexts();
   while(sessionContexts.hasMoreElements()) {
       SessionContext sc = (SessionContext)sessionContexts.nextElement();
%>
<hr>
<table width=600><tr><td>
<%= sc.toHTML() %></UL>
</td></tr></table>
<hr>
<table width=600><tr><td>
<center><h3>Session Dump Internals</h3></center>
</td></tr></table>
<pre>
<%
    int session_count = 0;
    Enumeration enum = SessionContextMonitor.tableKeys(sc);
    while(enum.hasMoreElements()){
        try {
            String key = (String)enum.nextElement();
            SessionData data = (SessionData)
                SessionContextMonitor.tableGet(sc,key);
            session_count ++;
            out.println(
                "<b>" + (session_count) +
                "</b>   " + data.getId());
            out.println("create time : " +
             df.format(new java.util.Date(data.getCreationTime())));
            long l = data.getLastAccessedTime();
            if ( l != -1L )
                out.println("last access : " +
                    df.format(new java.util.Date(l)));
            else
              out.println("last access : " +
                df.format(new java.util.Date(data.getCreationTime())));
            out.println("max inactive interval : " +
                data.getMaxInactiveInterval());
            out.println("user name : " + data.getUserName());
            out.println("valid session : " + data.isValid());
            out.print("new session : ");
            try { out.println(data.isNew()); }
            catch(Exception e){out.println(e.toString());}
            out.println("overflowed : " + data.isOverflow());

            //Cookie cookie = data.getCookie();
            //if ( cookie != null )
            //    out.println("cookie : " + cookie.toString());

            out.print("data : {");
            Enumeration s_keys = data.getNames();
            for(boolean first = true; s_keys.hasMoreElements(); ){
                if ( first )  first = false;
                else out.print(',');
                try {
                    String k = (String)s_keys.nextElement();
                    Object value = data.getValue(k);
                    out.print(k + "=" + value.toString());
                }catch(Exception e){out.println(e.toString());}
            }
            out.println('}');
            out.println();

            out.println("--------------------");
            //out.println(data);
        }catch(Exception e){ out.println(e.toString()); }
    }
    out.println("Total Http Session Count : " + session_count + "</b></pre>");
   } // end of while for each WebModules
}
catch(Exception e){
   out.println(e.toString() + "</pre>");
}
%>
<BR><hr>NOTE: This is only for debugging<br></body></html>
```

C.1 Signal information

Table C-1 Signal table

Signal #	name	comment
1	SIGHUP	Hangup (POSIX)
2	SIGINT	Interrupt (ANSI)
3	SIGQUIT	Quit (POSIX)
4	SIGILL	Illegal instruction (ANSI)

Signal #	name	comment
5	SIGTRAP	Trace trap (POSIX)
6	SIGABRT SIGIOT	Abort (ANSI) IOT trap (4.2 BSD)
7	SIGBUS	BUS error (4.2 BSD)
8	SIGFPE	Floating-point exception (ANSI)
9	SIGKILL	Kill, unblockable (POSIX)
10	SIGUSR1	User-defined signal 1 (POSIX)
11	SIGSEGV	Segmentation violation (ANSI)
12	SIGUSR2	User-defined signal 2 (POSIX)
13	SIGPIPE	Broken pipe (POSIX)
14	SIGALRM	Alarm clock (POSIX)
15	SIGTERM	Termination (ANSI)
16	SIGSTKFLT	Stack fault
17	SIGCHLD SIGCLD	Child status has changed (POSIX) Same as SIGCHLD (System V)
18	SIGCONT	Continue (POSIX)
19	SIGSTOP	Stop, unblockable (POSIX)
20	SIGTSTP	Keyboard stop (POSIX)
21	SIGTTIN	Background read from TTY (POSIX)
22	SIGTTOU	Background write to TTY (POSIX)
23	SIGURG	Urgent condition on socket (4.2 BSD)
24	SIGXCPU	CPU limit exceeded (4.2 BSD)
25	SIGXFSZ	File size limit exceeded (4.2 BSD)
26	SIGVTALRM	Virtual alarm clock (4.2 BSD)
27	SIGPROF	Profiling alarm clock (4.2 BSD)
28	SIGWINCH	Window size change (4.3 BSD, Sun)
29	SIGIO SIGPOLL	I/O now possible (4.2 BSD) Pollable event occurred (System V)

Signal #	name	comment
30	SIGPWR	Power failure restart (System V)
31	SIGUNUSED	

D

Webbank application overview

To illustrate how to work with the WebSphere Application Server, we use an application called Webbank. This application, which was originally developed for teaching purposes, allows you transfer money between a branch and a customer account and to check account balances. It is a full J2EE 1.3 application, using servlets, JSP, and enterprise beans. It also comprises two J2EE fat clients. This is obviously not a real-world application.

In this appendix, we describe what the Webbank application does and how it was written. We also explain in more detail some of the critical design points of this application.

D.1 Webbank design overview

This application has been written following most of the best practices for J2EE development and design, namely:

- Model-View-Controller architecture.
- Session pattern:
  - Entity beans are always accessed through session beans.
  - Clients (that is servlets) are shielded from the complexity of using EJBs.
- Business Interface pattern: definition of a common business interface that decouples definition of business methods from their implementation.
- Value objects (also known as data transfer objects) that are used to transport data between the client and the EJB layer and vice versa.
- Use of EJB 2.0 local interfaces.
- Use of JavaScript for basic form validation.

This application also uses the WebSphere Application Server tracing and logging mechanisms (JRas API). The different patterns used are described in more detail in the following sections.

D.2 Webbank application

The Webbank application provides two basic functions:

- View the branch and customer account balances.
- Transfer money between accounts.

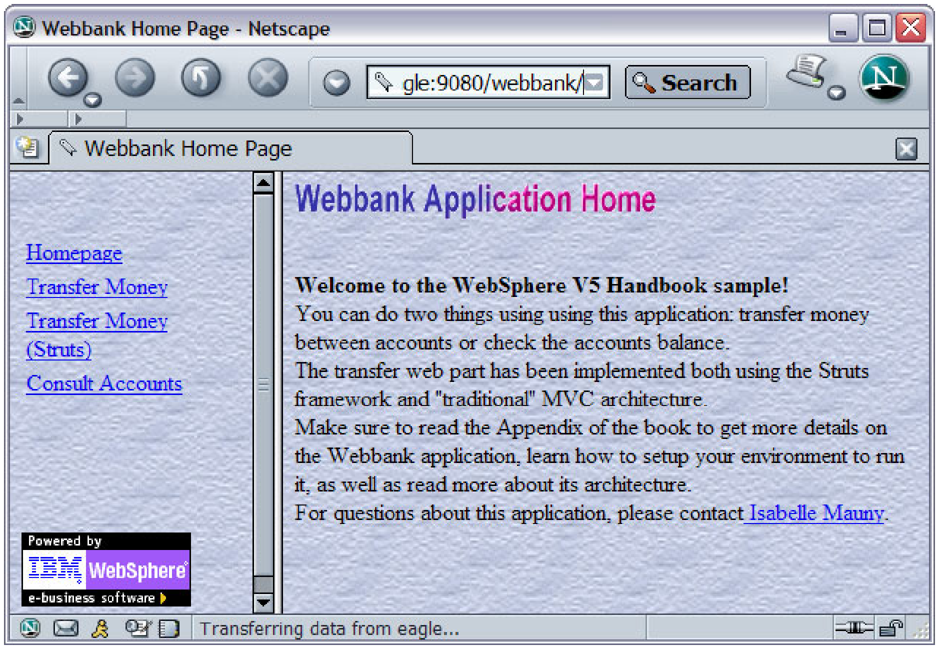
The same functionality is available via Web or via J2EE clients.

You can reach the Webbank application by going to `http://<hostname:port>/webbank/index.html`. The resulting page is shown in Figure D-1 on page 919.

If you want to make a transfer, you have two choices:

- Use the "traditional" implementation.
- Use the Struts-based implementation.

Both do exactly the same thing, and are meant to illustrate how to use various technologies.


Webbank Home Page - Netscape
gle:9080/webbank/
Search
Webbank Home Page
Homepage
Transfer Money
Transfer Money (Struts)
Consult Accounts
Powered by IBM WebSphere e-business software
Transferring data from eagle...

Webbank Application Home

Welcome to the WebSphere V5 Handbook sample!
You can do two things using using this application: transfer money between accounts or check the accounts balance.
The transfer web part has been implemented both using the Struts framework and "traditional" MVC architecture.
Make sure to read the Appendix of the book to get more details on the Webbank application, learn how to setup your environment to run it, as well as read more about its architecture.
For questions about this application, please contact Isabelle Mauny.

Figure D-1 Webbank application home page

D.2.1 Transfer servlet

Figure D-2 shows the basic flow of the Transfer servlet.

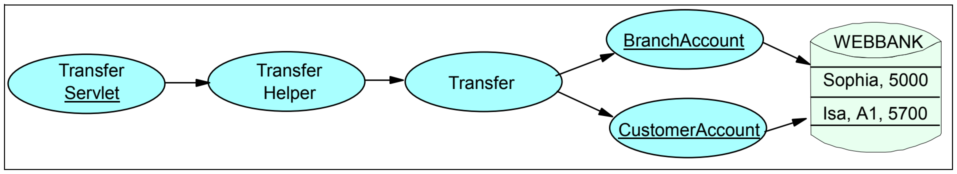


Figure D-2 Transfer servlet invocation flow

- The Transfer servlet gathers the transfer information from an HTTP form, and invokes the makeTransfer(TransferDTO) method of the TransferHelper class.
- The TransferHelper class accesses the Transfer session bean transparent to the servlet. The TransferDTO class is a value object, a JavaBean used to transport the data coming from the HTTP form back to the enterprise beans layer.
- The BranchAccount entity bean represents a branch account. Its primary key is the branch office name (such as Sophia).

- The Customer entity bean represents a customer. Its primary key is the customer name (such as Isabelle). The CustomerAccount entity bean represents a customer account. Its primary key is the accountID (such as A1). The Customer and CustomerAccount entity beans are linked by a one-to-many relationship (a customer can have multiple accounts).
- Entity beans persistence is managed by the EJB container (CMP).

D.2.2 Consultation servlet

The Consultation servlet gathers account information from a form, and simply displays the balance of each account. The invocation flow is basically the same as for the Transfer servlet, except that this flow uses a different session EJB, called Consultation.

D.2.3 Asynchronous transfers

A transfer happens on reception of a message. A message-driven bean (TransferMDB) is listening on the Webbank queue. Upon reception of a message, it uses the TransferHelper object to make a transfer.

You can test both synchronous and asynchronous transfers from the main transfer page.

D.2.4 J2EE clients

You can also use client applications to get account information and make transfers between accounts. Both clients are J2EE clients, that is they need a J2EE container to run (they use EJB references).

D.3 Implementation information

This section provides detailed information about the Webbank application implementation.

D.3.1 Using session facades

The Facade pattern is widely advocated as one of the pillar patterns in OO design, whatever the language. Its primary purposes, as stated in *Design Patterns: Elements of Reusable Object-Oriented Design*, are as follows:

- Use the Facade pattern when you want to provide a simple interface to a complex subsystem.

- Use the Facade pattern when you want to layer your subsystems. The facade will be the entry point to each subsystem level.

This leads to “layered development”. Each time you are entering a complex part of the system, you want to build a facade object that will:

- Isolate the previous layer from changes in the next layer. For example, if you consider a servlet using EJB sessions, you want to insert a facade layer to make sure that the servlet is independent from changes in the underlying technology (for example, EJBs).
- Shield the previous layer from the complexity of accessing the next layer. Again, if a servlet needs to access an EJB, it should not deal with JNDI lookup code, finding EJB homes, and so on. It should just call a method on the facade object, which in turn handles all the access logic to the session EJBs.

D.3.1.1 Session beans as facades

In the EJB world, session beans are also facades. Session beans are used to access the next layer, the persistence layer, most probably implemented with entity beans. The general rule is that clients (that is servlets/J2EE clients) should never access entity beans directly for three primary reasons:

- There is a significant runtime performance penalty that can be incurred when crossing the network on an RMI/IIOP call. If a client calls an entity bean representing two pieces of data (say the a flight number and the departing airport) that would require two network calls. When this is multiplied by a large number of attributes, this overhead quickly becomes significant.
- Each call to retrieve an attribute (that is each getXXXX() call) will begin a transaction, an SQL SELECT statement to synchronize the entity bean instance attributes with the database, eventually an SQL UPDATE statement (if you are not using EJB 2.0, or have not marked the getXXX() method as being a read-only method), and then a transaction commit. By using a session bean in front of your entity beans, you control the transaction demarcation. Session beans are used to drive the transaction. In other words, you will start one transaction, synchronize the entity bean attributes with the underlying database once, retrieve all the data, and commit the transaction.
- If you allow EJB clients to directly access an entity bean, it requires knowledge of the entity bean implementation that goes beyond what clients should have. For instance, manipulating an entity bean requires knowledge of the entity relationships (associations, inheritance) that are involved, inappropriately exposing the client to all of the details of the business model.

D.3.2 Using value objects

Value objects, also known as data transfer objects, are used to transport data from the server side (entity EJB layer) to the client side of an application, and vice versa. They are implemented as serializable Java beans. Usually, value objects are used as return types of session EJB methods, for example, AccountData getAccountInformation(). They contain a subset of entity beans attributes.

Consider centralizing the creation of value objects in a factory object. Since the list of attributes contained is subject to change, you will avoid having to make multiple changes in your code.

D.3.3 Using local interfaces

New with EJB 2.0, local interfaces are the cornerstone of container-managed relationships. By using local interfaces, you tell the container that the client calling the EJB via this interface is local (that is in the same JVM). This lets the EJB container perform a certain number of optimizations, the major one being passing data by reference and not by value (a great gain for RMI over IIOP calls).

D.3.4 Using a stateful session EJB

Stateful EJBs are rarely used in applications. Most of the time, you want to use stateless EJBs for scalability and performance. Using a stateful EJB for this application finds its roots in the original purpose: teaching EJB transactions. We used a stateful EJB to be able to use bean-managed transactions and drive the transaction from a graphical client. When you use the J2EE Transfer client, you can begin/rollback/commit a user transaction from the GUI. This is possible *only* by using a stateful EJB. We thought we would leave this implementation for information/teaching purposes. However, we did not use this implementation for the Web clients, as it is not justified at all. Instead, we used a stateless EJB.

D.4 Installing the Webbank application database

Two versions of the Webbank application are available with this book. One uses DB2 as its repository, the other one uses Cloudscape. In the following sections, you will find instructions to install each of them.

D.4.1 Instructions for DB2

All entity EJBs are persistent in a database called WEBBANK. Make sure you are logged in with an ID that is less than 8 characters long, and that has DBA rights, for example db2admin. Under UNIX systems, you must use a DB2 instance user ID, such as db2inst1.

Tip: If you are not logged on your Windows system with a correct DB2 user, you can still attach to a DB2 instance from a command prompt like this:

```
db2cmd
db2 attach to <instance_name> user <user> using <password>
```

For example:

```
dbcmd
db2 attach to DB2 user db2admin using <password>.
```

1. Start a command prompt, and start a DB2 command window:

   ```
   db2cmd
   ```

   Perform all actions below from this DB2 command window.
2. Create the WEBBANK database:

   ```
   db2 create db webbank
   ```
3. Create the WEBBANK tables:

   ```
   db2 connect to webbank user <username> using <password>
   db2 -tvf db2\tables.ddl
   ```
4. Populate the WEBBANK tables with some records:

   ```
   db2 -tvf db2\webbank.cli
   ```

D.4.2 Instructions for Cloudscape

If you want to simply test the Webbank sample without installing DB2 or Oracle, we provide a version of the application that works on top of Cloudscape V5. Cloudscape is a lightweight database, not meant to be used in production, that comes with the WebSphere Application Server. Most WebSphere samples, such as “Plants by WebSphere”, use Cloudscape as their persistent data store.

Note: Cloudscape V5.1 provides the following two separate frameworks for running Cloudscape with WebSphere Application Server:

- Embedded: this framework is the same as the one for Cloudscape V5.0. To use this framework, define a Cloudscape JDBC provider. See the Cloudscape section in the minimum required settings article for more information.

  You must use the embedded framework if you are running XA. Cloudscape does not support XA on Network Server.

- Network Server: this framework is a new feature in Cloudscape V5.1, and removes the limitations that existed in earlier versions of Cloudscape:
  - Inability to access a remote Cloudscape instance
  - Only one JVM can boot up the same database instance

This sample will use the embedded framework. For more information on using Cloudscape, see the InfoCenter.

Cloudscape tools setup

You will find two tools that you can use to administer Cloudscape V5 in the <WAS_HOME>/cloudscape/bin/embedded directory. Those two tools can be started using **cview.bat/.sh** and **ij.bat/sh**. Before using those tools, you should verify that the batch files correctly:

- Point to a specific Java runtime.
- Specify a list of JAR files needed to start those Java programs.

For example, we edited the cview batch file as follows. What we changed from the original version is in bold.

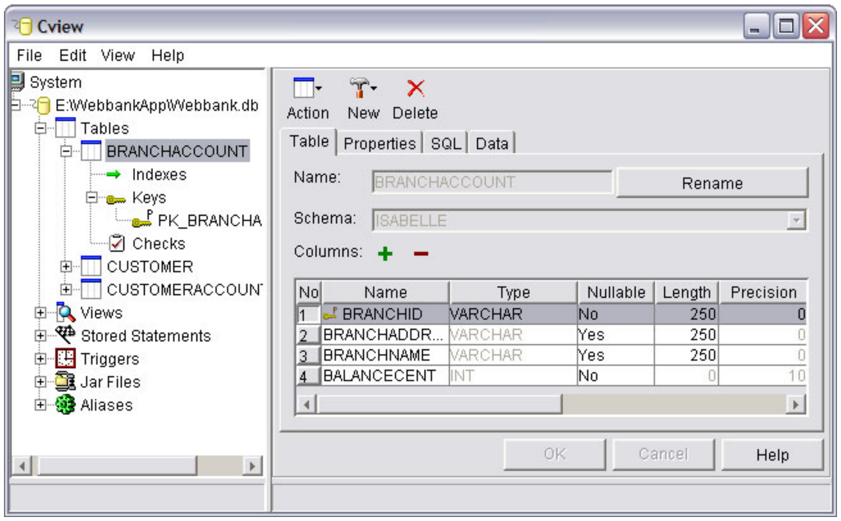
```
@REM ---------------------------------------------------------
@REM -- start cview
@REM ---------------------------------------------------------
..\java\bin\java -classpath
..\lib\db2j.jar;..\lib\db2jcview.jar;..\lib\db2jtools.jar;..\lib\jh.jar -ms32M
-mx32M com.ibm.db2j.tools.cview
```

Installing the Webbank Cloudscape database

The database comes in the form of a ZIP file that you have to unpack in a directory such as E:\WebbankApp. Assuming you unpack the Webbank.db.zip there, then your database name (as recognized by Cloudscape) is E:\WebbankApp\Webbank.db. Once you have unpacked the database, you can start the cview administration tool to make sure everything is OK.

1. Start cview.bat/cview.sh from the command line.

2. Invoke **File -> Open**, select the **E:\WebbankApp\Webbank.db** directory, and click **Open**. You should now see a window similar to Figure D-3 below. The database has been populated with some sample records, which you can see if you select the **Data** tab.



Name: BRANCHACCOUNT

Schema: ISABELLE

No	Name	Type	Nullable	Length	Precision
1	BRANCHID	VARCHAR	No	250	0
2	BRANCHADDR...	VARCHAR	Yes	250	0
3	BRANCHNAME	VARCHAR	Yes	250	0
4	BALANCECENT	INT	No	0	10

Figure D-3 Cloudscape viewer: Webbank database

D.5 WebSphere setup

The process of creating a JDBC provider and data source for Cloudscape is basically the same as for DB2. The only difference is in the values provided. The steps for DB2 are covered in 14.1.6, “Creating a DB2 JDBC provider and data source” on page 657. This section provides the information for creating the JDBC provider and data source for the sample webbank database if you are using Cloudscape.

D.5.1 Creating a Cloudscape JDBC provider

To create a JDBC provider for Cloudscape from the administrative console:

1. Expand the **Resources** entry and select **JDBC Providers**.
2. Select the scope of this resource. Select the server you are deploying to and click **Apply**.
3. Click **New**.

4. In the list of supported JDBC providers, select **Cloudscape JDBC Provider** and click **Apply**.
5. You then need to specify the following information:
   – Name

     The JDBC provider name, such as "WebbankCloudscapeProvider". For clarity, it is recommended that you include the name of the target database in this name.

   – Description

     An optional description of the JDBC provider.

   – Classpath

     The path to the JAR/ZIP file which contains the JDBC provider code, that is ${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar.

     Note: By default, the `CLOUDSCAPE_JDBC_DRIVER_PATH` variable is set to `{WAS_LIBS_DIR}`. This is correct since the Cloudscape libraries reside in <WAS_HOME>/lib. You can verify this setting by going to **Environment -> Manage WebSphere Variables.** Just make sure it has been set at the same scope or at a higher scope than the JDBC provider.

   – Implementation class

     The Java class that provides the JDBC service, that is com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource.
6. Click **Apply**.

D.5.2 Creating a Cloudscape data source

The next step is to create a data source that uses this JDBC provider. To create a data source, you need to navigate to the new JDBC provider entry (if you aren't already there from the previous step):

1. Click **Data Sources** in the Additional Properties table.
2. Click **New**.
3. The following properties can be specified for a data source:
   – Name

     The data source name, which must be unique in the domain. It is recommended that you use a value that indicates the name of the database this data source is used for, such as “WebbankDS”.

   - JNDI name

     This field must be set to “`webbank/jdbc/webbank`” for this application.
   - Description

     An optional (but recommended) description of this data source.
   - Container-managed persistence

     Check this option to indication that the Webbank data source is used for persisting the application entity beans.
   - Category

     A keyword, such as Applications or Samples, used to classify data sources. The list of data sources can be sorted using that field.
   - Statement Cache Size

     For better efficiency, WebSphere can maintain a cache of results of the ps.prepareStatement() call. Use this setting to set the size of this cache.
   - Datasource helper name

     The name of the class used by Relational Resource Adapter at runtime to implement the functionality of a specific database driver. This field is set automatically based on the JDBC provider, in this case to com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper.
   - Authentication aliases

     The authentication alias that defines access to the database. See 14.1.6, “Creating a DB2 JDBC provider and data source” on page 657.

4. Click **Apply** to create the data source.

Switch to the custom properties section to edit a list of properties specific to the database driver. When using Cloudscape, you must specify the database name property, that is E:/WebbankApp/Webbank.db.

Cloudscape tip: Cloudscape supports only one connection at a time. For example, if you have opened a Cloudscape database using cview, you can’t connect to it until you shut down the database.

D.5.3 Webbank tracing and logging support

Webbank has two tracing modes: you can either use the standard output file, or use the WebSphere logging and tracing support (JRas).

- If you want to send a trace to the standard out files, simply create a system variable called TEST_MODE and set it to true (-DTEST_MODE=true).

- If you want to activate the JRas trace, you can use the following trace specification:

  ```
  WebbankApp=all=enabled
  ```

E

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG246195`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com/redbooks`

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-6195.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
WebbankV51.ear	Webbank application deployed modules.
WebbankV51-WithSource.ear	Webbank application deployed modules, including the source code.
ClassloadersTest-WithSource.ear	Sample application for the classloaders example in 14.7, “Learning classloaders by example” on page 696.
ClassloadersTest-WithSource.jar	JAR file for the classloaders example in 14.7, “Learning classloaders by example” on page 696.
Table.ddl	SQL statements to create the DB2 database tables.
webbank.cli	SQL to populate the DB2 database tables.
Cloudscape.db.zip	Cloudscape database.

How to use the Web material

1. Read the description of the Webbank application in Appendix D, “Webbank application overview” on page 917.
2. Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.
3. Create and populate the WEBBANK sample database using the instructions in Appendix D.4, “Installing the Webbank application database” on page 922. Instructions are included for Cloudscape (included in WebSphere Application Server) and DB2.
4. Prepare the environment for the application by going step-by-step through 14.1, “Preparing the environment” on page 648.
5. Install the application using the instructions in 14.4, “Deploying the application” on page 677.

Opening the application in the Application Server Toolkit

Section 14.2, “Setting application bindings” on page 669 goes through the process required to define bindings for the application using the Assembly Toolkit functions provided in the Application Server Toolkit. The bindings discussed in this section have already been set for you, but if you would like to follow along, you can open the .ear file in the Application Server Toolkit by doing the following:

1. Open the Application Server Toolkit from <astk_install>astk.exe (.sh).
2. Open the J2EE perspective by selecting **Window -> Open Perspective -> Other -> J2EE** and then click **OK**.
3. Select **File -> Import -> EAR file** and click **Next**.
4. Browse to the location where you unzipped the SG246195.zip file. Select the **WebbankV51-WithSource.ear** file and click **Open**.
5. Click **Finish**.
6. For general information on using the Application Server Toolkit, see Appendix B, “Application Server Toolkit” on page 879.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 936.

- *IBM WebSphere V5.0 Performance, Scalability and High Availability,* SG24-6198
- *WebSphere Edge Server New Features and Functions in Version 2*, SG24-6511
- *IBM WebSphere Application Server - Express Handbook,* SG24-6555
- *WebSphere Commerce V5.4 Handbook*, SG24-6567
- *IBM WebSphere V5.0 Security Handbook,* SG24-6573
- *Patterns for the Edge of Network,* SG24-6822
- *Migrating to WebSphere V5.0: An End-to-End Migration Guide,* SG24-6910
- *Migrating WebLogic Applications to WebSphere V5,* REDP0448
- *IBM WebSphere V5.0 for Linux, Implementation and Deployment Guide*, REDP3601
- *WebSphere Installation, Configuration, and Administration in an iSeries Environment*, SG24-6588-00
- *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176

Other resources

These publications are also relevant as further information sources:

- *Java 2 Platform, Enterprise Edition Management Specification,* by Hans Hrasna, available at `http://java.sun.com/j2ee/tools/management`
- WebSphere Application Server V5 InfoCenter at `http://www.ibm.com/software/webservers/appserv/infocenter.html`
- *WebSphere MQ System Administration Guide*, SC33-1873-02

- *IBM WebSphere Application Server, Version 5 Getting Started*, available in both HTML and PDF form through the InfoCenter
- *Java Services Framework: An Approach to Open-Standards Service Development* (white paper)
- *An Introduction to JavaTM Stack Traces*, found at:

  `http://developer.java.sun.com/developer/technicalArticles/Programming/Stacktrace/`
- Apache HTTP Server log files, found at:

  `http://httpd.apache.org/docs/logs.html`
- Jim Gray, *Transaction Processing: Concepts and Techniques*
- Erich Gamma, Richard Helms, Ralph Johnson and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Design*. Addison-Wesley, 1995 ISBN 0-201-63361-2

Referenced Web sites

These Web sites are also relevant as further information sources:

- The IBM WebSphere Application Server home page

  `http://www.ibm.com/software/webservers/appserv/was`
- The IBM WebSphere Application Server Network Deployment home page

  `http://www.ibm.com/software/webservers/appserv/was/network/`
- The IBM WebSphere Application Server Enterprise home page

  `http://www-3.ibm.com/software/webservers/appserv/enterprise/`
- Sun's Web site

  `http://java.sun.com`
- The IBM WebSphere Application Server Express home page

  `http://www.ibm.com/software/webservers/appserv/express`
- J2EE Connector specification:

  `http://java.sun.com/j2ee/connector/`
- The Edge Components online InfoCenter

  `http://www-3.ibm.com/software/webservers/appserv/ecinfocenter.html`
- Patterns for e-business Web site

  `http://www.ibm.com/developerworks/patterns/`

- WebSphere Application Server supported release and prerequisites:

  `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`
- WebSphere MQ 5.2 MA88 SupportPac

  `http://www-4.ibm.com/software/ts/mqseries/txppacs/ma88.html`
- WebSphere MQ Integrator Web site

  `http://www-4.ibm.com/software/ts/mqseries/platforms/#integrator`
- WebSphere MQ 5.2 MA0C SupportPac

  `http://www-4.ibm.com/software/ts/mqseries/txppacs/ma0c.html`
- Solaris 8 support

  `http://sunsolve.sun.com`
- Sun ONE Web Server documentation

  `http://docs.sun.com`
- AIX maintenance

  `http://techsupport.services.ibm.com/server/nav?fetch=fdca`
- AIX maintenance

  `ftp://ftp.software.ibm.com/aix/fixes/51/ml`
- For more information on cookie properties, please refer to:

  `http://home.netscape.com/newsref/std/cookie_spec.html`
- API documentation

  `http://java.sun.com/j2ee/sdk_1.3/techdocs/api/`
- Thread Analyzer download

  `http://www7b.boulder.ibm.com/wsdd/downloads/ta.html`
- WebSphere Application Server support (FixPak, fixes, and hints and tips)

  `http://www-3.ibm.com/software/webservers/appserv/support.html`
- IBM alphaWorks emerging technologies

  `http://www.alphaworks.ibm.com`
- IBM developerWorks

  `http://www.ibm.com/developerworks/`
- Worldwide WebSphere User Group

  `http://www.websphere.org`
- Google search group: WebSphere Application Server

  `http://groups.google.com/groups?group=ibm.software.websphere.application-se
  rver`

- Jacl:

  `http://www.javasim.org/ref.script/tcljacl.htm`
- URL class documentation.

  `http://java.sun.com/j2se/1.3/docs/api/`
- URLs and their formats

  `http://java.sun.com/products/jdk/1.2/docs/api/java/net/URL.html`
- J2EE 1.3 documentation.

  `http://java.sun.com/j2ee/1.3/docs/`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

IBM

Redbooks

IBM WebSphere Application Server V5.1 System Management and Configuration

WebSphere Handbook Series

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

IBM WebSphere Application Server V5.1 System Management and Configuration
WebSphere Handbook Series

Exploring WebSphere Application Server V5.1's new features

Installing using popular platforms and technology

Mastering administration and application deployment

This IBM Redbook provides the knowledge needed to implement a WebSphere Application Server V5.1 Network Deployment runtime environment, to package and deploy Web applications, and to perform ongoing management of the WebSphere environment.

It is one in a series of handbooks designed to give you in-depth information on the entire range of WebSphere Application Server products.

This redbook provides an overview of the architecture, topology options, and features of WebSphere Application Server V5.1 and WebSphere Application Server Network Deployment V5.1.

It takes you through the installation steps needed to install each topology. Platform-specific chapters are included for installation on Windows, AIX, and Solaris.

The redbook then takes you through the process of configuring WebSphere Application Server. It also includes information about packaging and deploying applications, then concludes with information about troubleshooting runtime problems.

SG24-6195-01 ISBN 0738497630